Programación de Mainframes en Ensamblador

Bill Qualls

26 de marzo de 2010

Copyright © 2009 Bill Qualls – Derechos Reservados

Sobre el Autor Bill Qualls es actualmente profesor asistente de Matemáticas y ex profesor asistente de Sistemas de Información, en Waubonsee Community College, Sugar Grove, Illinois. Antes de unirse a Waubonsee trabajó en la industria durante nueve años como un programador de Ensamblador y COBOL. Bill obtuvo su B.A. en Contabilidad en Whittier College, su MBA de la Sistemas de Información de Keller Graduate School of Management, y su maestría en Matemáticas Aplicadas en De Paul University.

Dedicación Este libro está dedicado a mi madre Dorinda Qualls cuyo amor es incondicional. Me enseñó la división larga entre su paciencia y la caridad por su ejemplo.

Índice general

1. Primeros pasos con PC/370 **11**
- 1.1. Instalación . . . 12
- 1.2. Nuestro primer programa PC/370 . . . 13
- 1.3. Creación de conjuntos de datos de PC/370 . . . 17
- 1.4. Nuestro Primer Programa de Procesamiento de archivos . . . 23
- 1.5. Ejercicios . . . 33

2. Definición y movimiento de Carácteres **36**
- 2.1. Definir y movimiento de Datos de Carácter . . . 37
- 2.2. Definir y mover literales alfanuméricos . . . 47
  - 2.2.1. La Instrucción MVI . . . 52
  - 2.2.2. Valores equiparados . . . 55
- 2.3. Longitud explícita y el Desplazamiento . . . 58
- 2.4. Agregar y encabezados de columna . . . 63
- 2.5. Ejercicios . . . 69

3. IFS en BAL: Comparación de carácter FIELDS **74**
- 3.1. Compara en BAL: el código de condición . . . 75
- 3.2. Compara en BAL: La Instrucción CLC . . . 76
- 3.3. Compares in BAL: The CLI Instruction . . . 79
- 3.4. Compares in BAL:The BC Instruction . . . 80
- 3.5. Compares in BAL: Sample IFs . . . 81
- 3.6. Sample Program . . . 85
- 3.7. Exercicios . . . 88

4. Cómo estructurar un Programa de BAL **96**
- 4.1. CONFIGURACIÓN, PROCESO, y Wrapup Secciones . . . 97
  - 4.1.1. La Revisión de rutina PROCESO 103
  - 4.1.2. La Revisión de rutina Leer más 109
- 4.2. Ejercicios . . . 116

5. Representación de datos **118**
- 5.1. Una mirada más cercana a nuestro Sistema de numeración decimal . . . 118

5.2. El sistema binario Número 119
5.2.1. De binario a decimal de conversión 120
5.3. El EBCDIC y ASCII para colación de secuencias 120
5.3.1. Decimal a binario de conversión 122
5.3.2. Hexadecimal a decimal de conversión 123
5.3.3. Decimales a hexadecimal de conversión 124
5.3.4. Hexadecimal a binario de conversión 124
5.3.5. De binario a hexadecimal de conversión 124
5.4. Aritmética binaria 126
5.5. Aritmética hexadecimal 126
5.6. Ejercicios 127

6. ¿Qué es esta cosa de la izquierda? 130
6.1. El formato de instrucción SI 133
6.2. El formato de instrucción SS 134
6.3. El formato de instrucción RX 136
6.4. Instrucciones de Definición de Datos 137
6.5. Ejercicios 138

7. Bolsas de aritmética decimal 144
7.1. Presentación de Dindón Industrias 145
7.2. Bolsas de Formato Decimal 146
7.3. La Instrucción AP 150
7.4. La Instrucción ZAP 150
7.5. La detección de una condición de desbordamiento 151
7.6. La Instrucción UNPK 152
7.7. La Instrucción MVZ 152
7.8. Ejemplo de programación: La producción de la Crónica de Ventas 154
7.9. Ejemplo de programación: La producción de las discrepancias Informe de Inventario 161
7.10. La Instrucción SP 161
7.11. La Instrucción CP 162
7.12. Ejemplo de programación Revisited 163
7.13. El SS formato de instrucción Revisited 170
7.14. Datos de errores de excepción 171
7.15. Ejercicios 176

8. Lógica de Salto de página 182
8.1. La rutina CHKLNS 183
8.2. Incluyendo el número de página en las líneas de definición 184
8.2.1. La rutina HDGS 185
8.2.2. La Línea FORMFEED 185
8.2.3. Ejemplo de programa de solución 187
8.2.4. Ahorrar papel 191
8.3. Ejercicios 191

9. La Instrucción Editar **194**
9.0.1. Problemas con UNPK y MVZ Revistited 195
9.0.2. Definición de la máscara de edición 196
9.0.3. Indicando Importancia 198
9.0.4. Impresión Decimal Puntos 198
9.0.5. Impresión de la Señal 199
9.0.6. Impresión CR (crédito) o DB (de débito) 200
9.0.7. Hora de Protección 201
9.0.8. La documentación de la máscara de edición 201
9.1. Muestra del Programa 202
9.2. Informes Dindón Revisited: 205
9.2.1. Resumen 216
9.3. Ejercicios 217

10.Quiebre de la lógica de control de **221**
10.1. Single Level Control Break - Problema de programación 227
10.2. Single Level Control Break - Lógica 228
10.3. Single Level Control Break - Program Solution 232
10.4. Romper el control de nivel múltiple - Problema de programación 237
10.4.1. Romper el control de nivel múltiple - Lógica 237
10.4.2. La producción de OFFER10C.MLC - Ejemplo de nivel múltiple romper el control de 237
10.4.3. Romper el control de nivel múltiple - Programa de Solución 241
10.4.4. Totales Rolling 247
10.5. Informe resumido - Problema de programación 248
10.5.1. Informe resumido - Lógica 249
10.5.2. Informe resumido - Solución de programa 249
10.6. Ejercicios 255

11.Más de una entrada de archivo **258**
11.1. Clasificación de los archivos de entrada 259
11.2. La estructura de la línea principal 260
11.3. La rutina de instalación 261
11.4. Las rutinas de LEA 261
11.5. El proceso de rutina PROCESO 262
11.5.1. El CHKLNS, HDGS, y rutinas de escritura 264
11.6. La rutina Wrapup 265
11.6.1. La salida del programa 266
11.7. Programa de Solución 266
11.8. Ejercicios 273

12.cuencial del archivo de actualización 276
12.1. La estructura de la línea principal 280
12.1.1. La rutina de instalación 281
12.2. Las rutinas LEA 281
12.2.0.1. La rutina PROCESO 282
12.2.0.2. Master Low 282
12.2.1. transacción bajos 283
12.2.1.1. Válido Añadir 284
12.2.2. No válido el cambio, no se puede eliminar 284
12.2.2.1. Bad Código de transacción 285
12.2.2.2. Master / transacción del partido 285
12.2.3. Añadir malo 286
12.2.4. Válido Eliminar 286
12.2.5. Válido Cambio 287
12.2.6. Escribir nuevo Master 288
12.2.7. La rutina Wrapup 288
12.2.8. Programa de Solución 289
12.2.9. Conciliar el informe de auditoría 302
12.3. Ejercicios 303

13.Más Bolsas de aritmética decimal 307
13.1. Las instrucciones que Multiplicar 308
13.2. Programa de ejemplo: las ventas en dólares a nivel nacional Dindón's313
13.3. Redondeo con SRP 317
13.4. Muestra del Programa: Contribución de California a las ventas de 320
13.4.1. La Instrucción DP 320
13.4.2. Comentarios 325
13.5. Un ejemplo detallado 334
13.6. Solución 334
13.7. Ejercicios 336

14.Binary Arithmetic 340
14.1. Comparación de dos campos binarios 345
14.2. Mover un registro a una Fullword o halfword 347
14.2.1. Además Binario 348
14.3. Binario Resta 349
14.4. La conversión de Bolsas a Binary (CVB) 349
14.5. Conversión de binario a Bolsas (ECV) 350
14.6. Muestra del Programa - Datos Dindón Convertir a binario 351
14.7. Muestra del Programa - Discrepancias Informe de Inventario del archivo binario 356
14.8. Viewing a Register While Testing 361
14.9. Ejercicios 365

15.Tabla de Procesamiento **369**
15.1. Definición de una tabla; Sentinel ciclos controlados 370
15.2. Contador de ciclos controlados 373
15.3. Cargando una tabla en tiempo de ejecución 375
15.4. Los campos de tablas: El procesamiento de un campo de carácter por carácter 378
15.5. Muestra del Programa - la producción de transcripciones 382
15.5.0.1. Línea principal de la sección 389
15.6. La rutina de instalación 390
15.6.1. Carga de la tabla de golf 390
15.6.2. Las rutinas de LEA 391
15.6.3. La rutina PROCESO 392
15.6.4. GET Información del Curso de rutina 393
15.6.5. OBTENER GRADO DE VALOR rutina 393
15.6.6. DO GPA de rutina 394
15.7. La rutina Wrapup 395
15.7.1. La automatización del proceso con un archivo por lotes . 405
15.8. Un resumen de las más comunes Cuadro Instrucciones de procesamiento 405
15.9. Ejercicios 406

16.Más aritmética binaria **411**
16.1. Par-impar Registro de Pares 411
16.2. Binary Multiplicación: La M, MH, MR, y las instrucciones . . . 412
16.3. Binary División: El D y DR Instrucciones 415
16.4. Programa de ejemplo: las ventas en dólares a nivel nacional Dindón's 418
16.5. Muestra del Programa: Contribución de California a las ventas de 421
16.6. Ejercicios 425

17.A nivel de bits de Operaciones **429**
17.1. La O, Y, y exclusiva o de operaciones 430
17.2. El OI, NI, y XI de Instrucciones 430
17.3. The OC, Carolina del Norte, y XC Instrucciones 433
17.4. Manipulación de los registros: El SLL y SRL Instrucciones de . . 434
17.5. Programa de ejemplo: a nivel de bits de Operaciones 435
17.6. Programa de ejemplo: Acceso a la fecha y la hora 440
17.7. Bits de cheques: La Instrucción TM 443
17.8. Ejercicios 445

18.Indices **448**

Introducción

En el verano de 1992 mi escuela hizo lo que tantas escuelas estaban haciendo: quitaron sus computadoras centrales. Puedo entender totalmente sus razones para hacerlo: todos los trámites administrativos habían sido trasladado a otra plataforma, y esta máquina se mantenía exclusivamente para nuestras clases de programación en ENSAMBLADOR y en COBOL avanzado. Simplemente no era costo-efectivo el mantener la máquina. No obstante, me comprometi a seguir enseñando programación de ENSAMBLADOR para MAINFRAMES. Pero ahora parece como si estuviera sin un medio para hacerlo.

Varios años antes había oído hablar de PC/370, un programa shareware que permite al usuario escribir y ejecutar código en ensamblador de MAINFRAME en la PC. Escéptico, lo intenté. Y me encantó! Las diferencias entre el ENSAMBLADOR de MAINFRAME y PC/370 son triviales. PC/370 tiene la apariencia de la computadora central. Ni siquiera necesita una PC especial para ejecutar PC/370: cada programa en este libro fue escrito en un Compaq 286LTE utilizando DOS 3.4!

He utilizado PC/370 en mis clases de ensamblador en la escuela y en la industria desde entonces y nunca me ha decepcionado. El único problema que tenía era la falta de un libro de texto para apoyar la enseñanza usando PC/370. Ese fue el impulso original para escribir este libro.

Pero yo quería que este libro fuese algo más que una guía del usuario para PC/370. Yo quería escribir un texto fácil de leer sobre ENSAMBLADOR que haría hincapié en las aplicaciones de negocio, basado en mi experiencia como programador de ENSAMBLADOR en la industria. Creo que he tenido éxito en hacerlo.

En la actualidad, existe un renovado interés en la programación de MAINFRAME. Esto se debe principalmente a la campaña del Problema del año 2000 (también conocido como "Y2K" y "el error milenario"). Muchos "sistemas legados" fueron escritos en ENSAMBLADOR, y simplemente no hay el tiempo y/o dinero para volver a escribirlos en otro idioma. Pero deben hacerse compatible con el siglo. Aún más de estos sistemas están escritos en COBOL, y yo me suscribo a la opinión compartida por muchos en esta industria, de que, el aprendizaje de ENSAMBLADOR lo convertirá en un mejor programador de COBOL. El conocimiento de ENSAMBLADOR desmistifica el compilador de COBOL que es, después de todo, sólo un programador de ENSAMBLADOR.

Me doy cuenta de que solo algunos de los lectores de este texto se convertirá en los programadores de ENSAMBLADOR. Pero las habilidades aprendidas al

completar los ejercicios en este texto beneficiará al lector, independientemente de la lenguaje utilizado.

Este texto está escrito en un estilo no muy técnico. Escribirás programas en ENSAMBLADOR desde el primer capítulo. Ejecutarás los programas de ejemplo, completaras los ejercicios y te divertirás mientras aprendes!

Cómo está organizado este libro En el capítulo 1 podrás aprender la mecánica de la escritura, el montaje y ejecución de programas de ENSAMBLADOR que utilizan PC/370. También aprenderás cómo crear las bases de datos utilizadas en este texto.

En el capítulo 2, aprenderás los fundamentos de la definición y como mover campos de caracteres. A continuación, seras capaz de producir informes en formatos simples.

En el capítulo 3 aprenderás cómo hacer condiciones en ENSAMBLADOR. Probablemente será diferente a todo lo que has hecho antes. Pero puedes hacerlo: basta pensar a un nivel muy bajo!

En el capítulo 4, aprenderás cómo estructurar un programa en ENSAMBLADOR. Muchos programadores piensan "estructura" y "ENSAMBLADOR" no pertenecen a la misma frase. Pero si sigues algunas reglas muy simples, pronto escribiras programas que son muy fáciles de modificar y mantener.

En el capítulo 5 aprenderás más acerca de la representación de datos, incluyendo los sistemas de numeración binario y hexadecimal y las secuencias de caracteres EBCDIC y ASCII. Muchos textos de ensamblador presentan esta información en el primer capítulo, pero tuve que posponerlo hasta este capitulo en favor de permitir escribir programas tan pronto como sea posible.

En el capítulo 6 podrás aprender sobre algunos de los formatos de instrucciones de ENSAMBLADOR. Al término de este capítulo podras "ser" realmente el ENSAMBLADOR, determinando las instrucciones de código objeto que sería producido por el ENSAMBLADOR a partir del código fuente.

En el capítulo 7 usted aprenderá sobre aritmética decimal enpaquetada. De hecho la mayoría de la aritmética en ensamblador es decimal enpaquetados o números binarios. Los números decimales enpaquetados son los más importantes de los dos ya que se debe convertir a decimales enpaquetados para imprimir. Así que este es un capítulo muy importante.

En el capítulo 8, aprenderá acerca de la lógica de salto de página. Esta es una de las muchas aplicaciones de la aritmética que aprendimos en el capítulo 7.

En el capítulo 9, aprenderá acerca de la instrucción EDIT, que le permite imprimir un número decimal enpaquetado en casi cualquier forma que necesites (por ejemplo, con comas y punto decimal). En este punto será posible la elaboración de informes de calibre profesional.

En el capítulo 10 (Lógica .....), el capítulo 11 (más de un archivo de entrada), y el capítulo 12 (Numeración secuencial de archivos de actualización) podrá aprender la lógica necesaria para completar algunas aplicaciones de negocio muy comunes. No hay instrucciones de ENSAMBLADOR nuevas aquí: el énfasis

está en la lógica.

En el capítulo 13 aprenderás más acerca de aritmética decimal enpaquetada, específicamente cómo multiplicar y dividir. División en ENSAMBLADOR es particularmente rara, pero voy a mostrar algunos trucos absolutamente a prueba de tontos para hacerlo.

En el capítulo 14, aprenderás sobre numeros binarios o la aritmética de registros. En concreto, usted aprenderá Acerca de la suma y resta. Las cosas comienzan a ponerse un poco más confuso aquí, pero la recompensa hace que valga la pena. Este capítulo es especialmente importante para las ciencias de la computación mayores (información sistemas).

En el capítulo 15 aprenderás acerca del procesamiento usando tablas. El Procesamiento usando tablas no se podía hacer en ensamblador sin aritmética de registros.

En el capítulo 16 aprenderás más sobre binarios, o la aritmética de registros. En concreto, se aprender a multiplicar y dividir.

En el capítulo 17 aprenderás sobre las operaciones a nivel de bit. Hay algunas aplicaciones interesantes en de este capítulo, incluyendo el cambio de mayúsculas a minúsculas (y viceversa), la eliminación del signo de un campo numérico, la codificación, intercambio de dos campos, la multiplicación y la la división por potencias de dos, y el acceso a la fecha y hora del sistema.

En el Apéndice A encontrarás una descripción de los conjuntos de datos utilizados en este texto.

En el apéndice B podrás aprender a convertir tus programas PC/370 para ejecutarse en el MAINFRAME.

Nota para los instructores Tal vez perdió usted su mainframe, así como yo perdí la mía. Usted reconoce el valor de los conocimientos de programación en ENSAMBLADOR para todos los programadores de MAINFRAME, pero no han sido capaces de dar su curso por falta de un MAINFRAME. Con este libro y PC/370 y están de vuelta en el negocio!

He descubierto que puedo abarcar todo el texto en un semestre en Waubonsee Community College. La mayoría de estos estudiantes ya han tenido un semestre de BASIC, un semestre de de lógica de programación, y un semestre de COBOL. Usualmente tengo prisa el final del semestre, pero se puede hacer!

Al planificar su curso, puede que desee mantener en mente lo siguiente:

Los Objetivos se encuentran al comienzo de cada capítulo y son muy beneficiosos para en la planificación de su curso.

La mayoría de los capítulos contienen "INTENTA" en el texto del capítulo. Por lo general el uso de estos en la clase, impulsan una buena discusión y me ayudan a mantenerme en la pista.

Me he desprendido de la representación de datos hasta el capítulo 5 en favor de que mis alumnos escriban programas de ensamblador lo antes posible. Me doy cuenta de que muchos profesores prefieren cubrir este material en la primera semana del curso. Este capítulo puede ser discutido antes, pero la discusión de los programas de ejemplo que figuran en él tendrá que ser aplazado.

El capítulo 10 (Lógica...), el capítulo 11 (más de un archivo de entrada), y el capítulo 12 (Numeración secuencial de archivos de actualización) discutiran las técnicas de programación se encuentran comúnmente en los negocios de aplicaciones. Creo que estos capítulos son muy importantes para el futuro de los programadores de aplicaciones, pero ya que ninguno de estos capítulos introducen nuevas instrucciones de ensamblador, y como estos conceptos pudo haber sido discutido en otros cursos, puedes saltar cualquier o todos ellos según sea necesario para adaptarse al tiempo disponible.

Las tareas de programación en este texto hacer uso de las "bases de datos" que figura en el Apéndice A. Usted notará el tema recurrente de "SMALL TOWN". Este se utiliza en la mayoría de los programas de ejemplo. Es posible que desee elegir uno o dos de las otras bases de datos, tales como la tienda de hardware de la pequeña ciudad o del Banco de Sangre de la pequeña ciudad y elegir los ejercicios de programación que utilizan estas bases de datos. De esta manera, la complejidad de los programas que sus alumnos desarrollarán estaran en proporcion a la duración del curso.

Ninguno de los campos de fechas en cualquiera de los archivos en este texto incluyen el siglo. Como usted probablemente sabe, este es usualmente el caso con los sistemas heredados. Reta a tus estudiantes a hacer sus programas "siglo conformantes".

Si sus estudiantes todavía tienen acceso a una computadora central, a considerar la posibilidad de desarrollar sus programas utilizando PC/370, a continuación, suban sus programas a la computadora central y hagan los cambios necesarios para ejecutar sus programas allí. Yo lo hice, durante la enseñanza de este curso en la Universidad De Paul y fue de mucho éxito. (Los cambios son triviales y se discuten en el Apéndice B.)

Acerca de PC/370 Al ensamblar y enlazar un programa PC/370, podrás ver un recordatorio de que es shareware y que usted debe enviar cuarenta y cinco dólares para convertirse en un usuario registrado. No es necesario registrarse, como se indica a continuación (carta en el archivo):

> Por su petición le envío la presente carta confirmando que, Donald S. Higgins, autor de la Paquete shareware PC/370 ya no acepto registros, y usted es libre para distribuir la versión shareware sin cargo alguno. La última versión del shareware PC/370 es de 4,2 de enero 1988. Hay una copia de esta versión disponible para descargar en el PC Compuserve en el Foro de programadores de la biblioteca bajo el nombre de PC370.ZIP.
>
> Atentamente,
> *(Firmado)* Donald S. Higgins

Gracias a Don para el desarrollo de este gran producto!

Capítulo 1

Primeros pasos con PC/370

Objetivos Al término de este capítulo usted será capaz de:

- Describir brevemente las generaciones de lenguajes de programación,
- Instalar PC/370,
- Editar, ensamblar, enlazar y ejecutar un programa simple,
- Crear un conjuntos de datos para su uso con PC/370, y
- Producir una lista formateada de los registros en un archivo.

Introducción PC/370 es un producto shareware desarrollado por Don Higgins, que permite al usuario escribir y ejecutar programas en lenguaje ensamblador para sistemas System/370 (MAINFRAME) en un ordenador personal. Antes de el desarrollo de PC/370, una MAINFRAME era necesaria para hacer esto.

El lenguaje de ensamblador System/370 se refiere a menudo como BAL por sus siglas en ingles -*Basic Assembler Language*-. BAL es un idioma de segunda generación. La primera generación de idiomas son los idiomas de las máquinas. Los programas en idiomas de las máquina fueron escritas en código binario (normalmente representado por unos y ceros.) Para ejemplo, el código de lenguaje de máquina necesaria para mover un campo a otro podría ser como sigue:

11010010000000111101000010101011110100001010000

Como se pueden imaginar, estos programas son muy difíciles de escribir y aún más difícil de mantener. El lenguaje de ensamblador fue desarrollado para superar este problema. La idea era asignar un código mnemónico simple para cada operación y sus operandos. Por ejemplo, los primeros ocho bits de la código de máquina que se ha indicado, 11010010, indica una operación de desplazamiento de caracteres, por lo que sería MVC, entonces utilizar en su lugar esa palabra: MVC es, después de todo, mucho más fácil de recordar que 11010010!

Otro programa, llamado el ensamblador, traduciría estos mnemónicos al código de máquina equivalente, ya que en última instancia, todos los programas deben estar a nivel de código máquina para que el equipo les entienda.

El problema con la codificación en ENSAMBLADOR es que todavía es codificación a nivel de máquina, y cada máquina tiene su propia lengua. En consecuencia, un programa escrito en lenguaje de máquina para un plataforma (como una unidad central) no se puede ejecutar en otra plataforma (como en una PC). Esta fue una parte del impulso para la siguiente generación de lenguajes, la tercera generación. Esto incluye la generación de FORTRAN, COBOL, BASIC, RPG y C. La intención es que un programador podría escribir el código fuente que sería en gran medida independiente de la plataforma en la cual el código se ejecutaría en. El código fuente a continuación, se ejecuta a través de un compilador que traduce el código fuente para el código de máquina apropiado para la plataforma destino. Sólo el compilador (no el código fuente) sería dependiente de la máquina.

Esta discusión plantea dos preguntas. En primer lugar, ¿cómo podemos ejecutar BAL en una PC, y en segundo lugar, ¿por qué queremos aprender BAL? PC/370 nos permite ejecutar BAL en una PC, ya que emula una mainframe. Que esto sea posible, no debería ser una sorpresa: la portabilidad es, después de todo, una función del traductor no una función del código fuente.

Hay varias razones para aprender BAL. En primer lugar, hay muchas aplicaciones heredadas en los negocios escrito en BAL. Sencillamente, no es económico volver a escribir estos programas en otro idioma. En vez de, mantenemos, y por ende debemos saber BAL para hacerlo. En segundo lugar, algunas aplicaciones son simplemente más fácil escribir en BAL. Como un profesional, cada lenguaje de programación es otra herramienta en su cinturón de herramientas. Como profesional, usted debe saber que herramienta a utilizar para que trabajo. Mientras COBOL, C, o incluso de LOTUS pueden ser la mejor opción para algunos trabajos, BAL es la lengua de elección para los demás. En tercer lugar, ya que cada lengua en última instancia, debe ser reducido a su lenguaje de máquina equivalente (de nuevo, esta es la función del compilador), y ya que BAL es más cercano al lenguaje de máquina, el aprendizaje BAL te hará un mejor programador, independientemente del idioma que codifiques en. Por último, algunos de nosotros diríamos que es divertido!

Para completar, se debe mencionar que los lenguajes de cuarta generación (comúnmente denominado 4GL[1]) no son lenguas de procedimiento sino de apoyo a consultas de base de datos y presentación de informes "ad hoc". Estos incluyen SQL, RAMIS, FOCUS, NOMAD, y otros.

1.1. Instalación

El disquete incluido con este texto tiene tres directorios: \FULL, \MIN, y \SOURCE. El directorio \FULL contiene el paquete PC/370 entero. (La mayoría de los autores del shareware requieren que su software sea distribuido en su

[1] 4th generation languages por sus siglas en ingles

totalidad.) Gran parte nunca lo usaras. Para simplificar el uso de PC/370, el Directorio \MIN contiene los archivos mínimos necesarios para escribir y ejecutar todos los programas contenidos en este texto.

Los programas pueden ser almacenados en y ejecutados desde un disquete. Si quieres hacer eso, entonces debe hacer una copia de seguridad del disquete de distribución. Esto se puede hacer con DISKCOPY desde el comando de DOS: desde el *C:\>* del sistema, escriba DISKCOPY A: A: y pulsa ENTER. Intercambiar discos cuando se indique. Mantenga el disquete de distribución en un lugar seguro. (Si se utiliza una unidad de doble densidad, Basta con copiar la carpeta \MIN desde el disquete de distribución a su disquete de trabajo.)

Si va a almacenar y ejecutar sus programas desde un disco duro, a continuación, cree un directorio (por ejemplo, como \PC370) y use los comandos de XCOPY de DOS para copiar todos los directorios y archivos desde el disquete al directorio. Por ejemplo,

```
C:\> md pc370
C:\> cd pc370
C:\PC370> xcopy a: *.* /s
```

A lo largo de este texto, se supone que PC/370 se ejecuta desde la directorio \MIN de un disquete.

El directorio \SOURCE contiene el código fuente de todos los ejemplos contenidos en este libro. Se le anima a montar y ejecutar estos programas. Para ello, debe copiarlos (uno a la vez) al directorio \MIN. Experimente con ellos. Al igual que toda otra lengua, así es como realmente se aprende!

1.2. Nuestro primer programa PC/370

El siguiente programa mostrará un mensaje en la pantalla. Utilice su editor preferido para crear HELLO.MLC como se muestra a continuación, a continuación, siga la secuencia de comandos en la página siguiente para comprobar que se ha PC/370 ha instalado correctamente.

13 ⟨*programa no.1, version completa* 13⟩≡

```
            PRINT NOGEN
 ***************************************************************
 *          FILENAME: HELLO.MLC                                *
 *          AUTHOR : Bill Qualls                               *
 *          SYSTEM : PC/370 R4.2                               *
 *          REMARKS : This program will display a message.     *
 ***************************************************************
            START 0
            REGS
 BEGIN      BEGIN
            WTO    MESSAGE
            RETURN
```

```
MESSAGE    DC     C'Hello world!'
           END    BEGIN
```

Cuidado! no escribas mas alla de la columna 71!

Al introducir este programa toma en cuenta lo siguiente:

- El programa se crea en el directorio \MIN.
- Un asterisco (*) en la primera columna indica que la línea es un comentario.
- Las etiquetas (como BEGIN y MESSAGE) siempre comienzan en la columna uno.
- Debe haber al menos un espacio en blanco entre las etiquetas y las operaciones: por convenio, las operaciones (tales como PRINT) comienzan en la columna de diez.
- Debe haber al menos un espacio en blanco entre las operaciones y operandos: por convención, los operandos (como NoGen) comienzan en la columna dieciséis.
- Las etiquetas, las operaciones y operandos deben estar en mayúsculas; comentarios pueden ir en mayúsculas o minúsculas.
- Todo el código no debe extenderse más allá de la columna 71.

El diagrama de flujo en la página 5 muestra las entradas y salidas para cada paso del proceso de montaje.

```
A:\> cd min
A:\MIN> m370 hello
A:\MIN> a370 hello/lx
        ********************************************
        *    PC/370 System Release 4.2       01/07/88 *
        *    Copyright (C) 1988 Donald S. Higgins     *
        *                                             *
        * You are encouraged to copy and share this   *
        * package with other users on the condition   *
        * the package is not distributed in modified  *
        * form, and that no fee is charged. If you    *
        * find PC/370 useful, send 45 dollars to the  *
        * address below to become registered user and *
        * support continued shareware development.    *
        * Registered users will receive notices of    *
        * future PC/370 releases.                     *
        *                                             *
        *      Don Higgins                            *
```

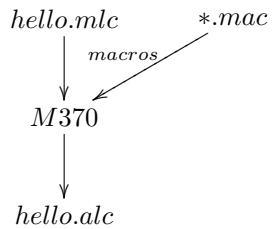
```
         *      6365 - 32 Avenue, North                *
         *      St. Petersburg, Florida 33710          *
         ********************************************
        PC/370 CROSS ASSEMBLER OPTIONS=LXACE
        STATS SYM=00021 MAXSTD=00005 LIT=00000 MAXLTD=00000 BTMEM=51900
        NO ERRORS FOUND
A:\MIN> l370 hello/lx
         ********************************************
         *    PC/370 System Release 4.2       01/07/88 *
         *    Copyright (C) 1988 Donald S. Higgins     *
         *                                             *
         * You are encouraged to copy and share this   *
         * package with other users on the condition   *
         * the package is not distributed in modified  *
         * form, and that no fee is charged. If you    *
         * find PC/370 useful, send 45 dollars to the  *
         * address below to become registered user and *
         * support continued shareware development.    *
         * Registered users will receive notices of    *
         * future PC/370 releases.                     *
         *                                             *
         *      Don Higgins                            *
         *      6365 - 32 Avenue, North                *
         *      St. Petersburg, Florida 33710          *
         ********************************************
        PC/370 LINKAGE EDITOR OPTIONS ON = LXEFIP
        STATS SYM=00002 MAXSTD=00001 BTMEM=57756
        NO ERRORS FOUND
A:\MIN> hello
         Hello world!
A:\MIN>
```

Note:

I have a letter on file from Mr. Don Higgins indicating that he no longer accepts registrations for PC/370. You are free to use this product without registration or fee. So disregard this message!

Thanks, Don!

1. Tecleas M370 HELLO



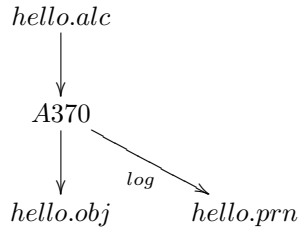
HELLO.MLC es su código fuente Use .MLC como la extensión.

Las librerias de macros se suministra con PC/370 en archivo con la extension .MAC

M370 es el prepocesador de macros (un programa que genera el archivo .ALC)

El archivo .ALC contiene su código fuente con macros expandidas; puede ser eliminado después de el archivo .OBJ ha sido ha sido creado.

2. Tecleas A370 HELLO/LX. *Nota: no hay espacios permitidos entre el nombre del programa, barra, diagonal y lx*

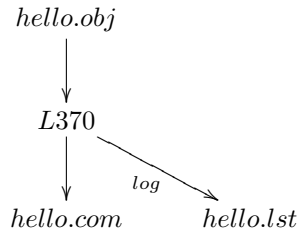


A370 es el ENSAMBLADOR.

El archivo .PRN contiene un resumen de lo que sucedio esto es, numera su código fuente, código objeto, y los errores de ENSAMBLADOR, si los hubiere.

El archivo .OBJ se va a usar en la ultima etapa.

3. Tecleas l370 hello/lx.



El archivo .OBJ puede ser eliminado despues que el archivo .COM ha sido creado.

L370 es el LINKER.

El Archivo .lst contiene los error del linker, si lo hubiere. Generalmente no es necesario.

El archivo .COM es lo que se ejecuta. Sin embargo, no trabaja solo; debe de encontrar E370R42.EXE en la ruta de acceso o vas a obtener el mensaje de error de PC/370: "NO ENCONTRADO"!

SID	SNAME	SSEX	SMAR
713	HILMER, D.R.	F	M
421	QUALLS, G.E.	M	S
701	ARIAS, I.L.	F	M
125	MORALES, L.A.	F	M
896	QUALLS, D.M.	F	S
626	MERCIER, J.L.	F	M
402	FOOTE, A.K.	F	M
263	HAVLIK, K.M.	M	M

Cuadro 1.1: STUDENT

CID	CDESC	CHRS
AC101	ACCOUNTING	3
BU101	BUSINESS	3
EG101	ENGLISH I	3
EG102	ENGLISH II	3
MA101	ALGEBRA	3
MA107	STATISTICS	3
PE121	FIRST AID	1
PE151	AEROBICS	1

Cuadro 1.2: COURSE

1.3. Creación de conjuntos de datos de PC/370

La mayoría de los programas mostrados en este texto se utilizan los archivos de la Comunidad "Small Town College". Esa base de datos se muestra aquí en su totalidad:

* * * * * * * * * * * * * * * * * * * *

TID	TNAME	TDEG	TTEN	TPHONE
732	BENSON, E.T.	PHD	N	5156
218	HINCKLEY, G.B.	MBA	N	5509
854	KIMBALL, S.W.	PHD	Y	5594
626	YOUNG, B.	MBA	Y	5664
574	SMITH, J.	MS	Y	5320

Cuadro 1.3: TEACHER

SEM	CID	SECT	TID	ROOM
W92	EG102	1	732	A1
W92	MA107	1	218	A2
W92	PE151	1	574	GYM
F92	AC101	1	218	B1
F92	BU101	1	218	B1
F92	EG101	1	732	A1
F92	EG101	2	732	A1
F92	MA101	1	626	A2
F92	MA101	2	626	A2
F92	PE151	1	574	GYM
W93	EG102	1	854	A1
W93	MA101	1	626	A2
W93	MA107	1	626	A3
W93	PE151	1	574	GYM

Cuadro 1.4: OFFER

SID	SEM	CID	SECT	GRADE
626	W92	EG102	1	A
896	W92	PE151	1	A
263	W92	PE151	1	C
896	F92	AC101	1	C
896	F92	BU101	1	C
896	F92	EG101	1	A
713	F92	EG101	2	C
421	F92	EG101	2	B
713	F92	MA101	1	B
896	F92	MA101	1	B
125	F92	MA101	2	F
701	F92	MA101	2	B
263	F92	PE151	1	B
701	F92	PE151	1	A
713	W93	EG102	1	B
421	W93	EG102	1	A
896	W93	EG102	1	B
125	W93	MA101	1	C
713	W93	MA107	1	B
896	W93	MA107	1	A
701	W93	MA107	1	D
263	W93	PE151	1	A

Cuadro 1.5: GRADE

Considere la siguiente tabla (archivo) de los profesores (tabla 1.3 TEACHERS) adoptadas por el Colegio Small Town de la Comunidad base de datos.

El diseño de registro siguientes describen el archivo TEACHER.DAT:

Nº	Nombre	Descripción	Inicio	Fin	Long	Formato
1	TID	ID Profesor	1	3	3	ZD
2	TNAME	Nombre del Profesor	4	18	15	CH
3	TDEG	Grado más alto	19	22	4	CH
4	TTEN	Titular?	23	23	1	S/N
5	TPHONE	Teléfono Profesor	24	27	4	ZD
6	TCRLF	Sólo pc/370	28	29	2	CR/LF

ZD — está dividido en zonas para decimal. Esto significa que el campo es numérico, sino que se almacena en un inmediato ZD formato legible (en lugar de relleno o binaria, que veremos más adelante.) Si usted sabe COBOL, este es el mismo que un campo PIC 9.

CH — representa el carácter. Esto significa que el campo puede o no puede ser una letra. Todo vale. En COBOL, este es el mismo como un campo de PIC X

CR/LF — significa un retorno de carro / avance de línea. Esta es una consideración muy importante cuando se utilizan CR/LF PC/370. En un entorno de mainframe, bases de datos generalmente utilizan registros de longitud fija, es decir, todos los registros de tienen el mismo número de caracteres, y cada campo de un registro empieza en la misma posición en todas las registros. Si su experiencia se limita a la programación BASIC, este probablemente será muy nuevo para usted.

Siempre que utilice un PC para crear un conjunto de datos como ésta, usted, naturalmente, pulsa RETURN después de cada registro o fila. En una PC, pulsando la tecla RETURN añade dos caracteres a un registro: retorno de carro (CR) y salto de línea (LF). Estos son los verdaderos caracteres, cada uno de los cuales ocupa un byte de memoria o de almacenamiento, como haría con cualquier otro carácter, como una 'A' o '*'. El CR/LF debe tomarse en cuenta en la determinación de la longitud de un registro cuando se utilizan PC/370.

En el siguiente ejemplo, el comando COPY de DOS "se utiliza para crear el archivo maestro en el directorio \MIN. CON es un nombre de dispositivo de DOS de la consola, o el teclado. Esta es una forma conveniente de crear pequeños conjuntos de datos utilizando DOS. Entradas de usuario se muestran en negrita. Tenga en cuenta que ^Z se lee como "control Z" y se introduce pulsando Z

mientras mantiene presionada la tecla Ctrl, o pulsando la tecla de función F6. Pulse RETURN después de cada línea, incluso después de ^Z

```
A:\> cd min
A:\MIN> copy con teacher.dat
732BENSON, E.T.   PHD N5156
218HINCKLEY, G.B. MBA N5509
854KIMBALL, S.W.  PHD Y5594
626YOUNG, B.      MBA Y5664
574SMITH, J.      MS  Y5320
^Z
1 File(s) copied
A:\MIN> dir teacher.dat
        Volume in drive A has no label Directory of A:\MIN
        TEACHER DAT 145 6-23-93 8:01a
        1 File(s) 932864 bytes free
A:\MIN> type teacher.dat
        732BENSON, E.T.   PHD N5156
        218HINCKLEY, G.B. MBA N5509
        854KIMBALL, S.W.  PHD Y5594
        626YOUNG, B.      MBA Y5664
        574SMITH, J.      MS  Y5320
```

Nota del comando DIR que el tamano total del archivo es de 145 bytes. ¿Por qué? Cada registro usa 27 bytes para los datos y 2 bytes para el CR/LF, para un total de 29 bytes por registro. Hay 5 registros en el archivo, y $5 * 29 = 145$ bytes. Esto demuestra que, como se dijo anteriormente, cuando se utiliza un PC, al pulsar el Tecla RETURN añade dos caracteres a un registro.

Hay una desventaja real de usar COPY con la creación de un conjunto de datos: si usted comete un error debe teclear ^Z y presione RETURN para finalizar, y luego comenzar de nuevo. Es mucho más fácil de usar un editor para crear un conjunto de datos.

ADVERTENCIA: algunos editores de PC quitan los espacios en blanco. Estos editores están muy bien para escribir código fuente de PC/370, pero no se puede utilizar para crear conjuntos de datos de PC/370. Si usted no está seguro de cómo su editor de obras, utiliza el comando DIR para comprobar el tamaño del archivo y comprobar que tal como lo hicimos anteriormente.

Si, después de pasar por el proceso anterior para validar el tamaño del archivo, usted encuentra que usted está fuera de el 1 de solo byte, no te preocupes. Su editor, probablemente añadido un especial de fin de archivo (EOF) a su archivo. Eso está bien.

He aquí otro ejemplo. Considere la tabla de cursos (Tabla 1.4 OFFER) que se ofrecen desde el pequeño pueblo Base de datos de la Universidad.

El diseño de registro siguientes se describe el archivo:

Nº	Nombre	Descripción	Inicio	Fin	Long	Formato
1	SEM	Semestre	1	3	3	CH
2	CID	ID Curso	4	8	5	CH
3	SECT	Sección	9	9	1	ZD
4	TID	ID Profesor	10	12	3	CH
5	ROOM	Cuarto	13	16	4	CH
6	OCRLF	Sólo pc/370	17	18	2	CR/LF

Observe que el número de sala se define como cuatro bytes de longitud, pero sus valores son sólo dos o tres bytes de longitud (por ejemplo, A1 o en el gimnasio). Desde ese ámbito se define como cuatro bytes de longitud, *cuatro bytes deben de ser teclados!* En el siguiente ejemplo, he pulsado la barra espaciadora dos veces después de cada habitación de dos bytes, y una vez después de cada gimnasio, antes de pulsar RETURN. Pruébelo usted mismo. Como puede ver, el total de número de bytes es 252. Esto se debe a $(16 + 2) * 14 = 18 * 14 = 252$.

```
A:\MIN> copy offer.dat con
        W92EG1021732A1
        W92MA1071218A2
        W92PE1511574GYM
        F92AC1011218B1
        F92BU1011218B1
        F92EG1011732A1
        F92EG1012732A1
        F92MA1011626A2
        F92MA1012626A2
        F92PE1511574GYM
        W93EG1021854A1
        W93MA1011626A2
        W93MA1071626A3
        W93PE1511574GYM
        ^Z
        1 archivo (s) copied
A:\MIN> dir offer.dat
        Volumen en la unidad A no tiene etiqueta
        Directorio de A:\min
        OFERTA DAT 252 6-23-93 10:25
        un 1 archivo (s) 899072 bytes libres
```

1.4. Nuestro Primer Programa de Procesamiento de archivos

Para nuestro primer programa de procesamiento de archivos, se producirá una lista de los registros en el archivo TEACHER.DAT. Esto programa leerá un registro, escribir un registro, sin formato en absoluto. Dicha lista es comúnmente denominada un 80/80 o la lista de tarjetas imagen. El término se remonta a los días de las tarjetas perforadas. Cada tarjeta perforada tenia 80 caracteres, y era común tener un programa que simplemente listaba las tarjetas en una cubierta. Eso es lo que nuestro programa va a hacer. Vamos a mejorarlo después.

Nuestro propósito al hacer este programa es introducir a los "generales" que se encontraran en todos nuestros programas. En muchos casos, no se explicará en detalle, como la PROCEDURE DIVISION no se explica en una clase COBOL de principiantes: nos limitaremos a señalarlo y decir cómo para usarlo sin entrar en los por qué en este punto.

El programa y sus resultados se muestran en la página siguiente. Aquí está nuestra breve introducción a estos comandos comunes. Hablaremos acerca de la mayoría de ellos en más detalle más adelante.

PRINT...NOGEN No imprimir las expansiones de macro. Eso no significa nada para ti ahora y, de hecho, no tiene ningún efecto en PC/370. Pero es algo que se encuentra al comienzo de casi todos los programas BAL, y que debe escribirse también.

* Asterisco en una columna indica que toda la línea es un comentario. Favor de usar * libremente.

START...0 Marca el inicio de su programa. Directivas de Assembler (como PRINT Start 0 NoGen) y los comentarios pueden hacerse antes de esta declaración. Eso es un cero después del inicio. Siempre incluya esta declaración.

23a ⟨*Prologo* 23a⟩≡ (28 30 32 45 61 67 72 86) 23b▷

```
START 0
```

REGS Una macro que asigna un nombre a cada registro (por ejemplo, para R6 REGs registro de seis). La necesidad de una macro de este tipo es tan común que un similar macro se encuentra en las tiendas más BAL. Puede ser llamado EQUATE, REGEQU, o algo similar. No es necesario, en este programa particular, pero se suele añadir a todos los programas por la costumbre, y su uso aquí sirve para corroborar que tiene PC/370 instalado correctamente.

23b ⟨*Prologo* 23a⟩+≡ (28 30 32 45 61 67 72 86) ◁23a

```
REGS
```

24 ⟨*TEACH1A.MLC, listado de maestros, version completa* 24⟩≡

```
          PRINT NOGEN
*****************************************************************
*         FILENAME: TEACH1A.MLC                                 *
*         AUTHOR : Bill Qualls                                  *
*         SYSTEM : PC/370 R4.2                                  *
*         REMARKS : A card-image list of teacher records.       *
*****************************************************************
          START 0
          REGS
BEGIN     BEGIN
          OI     TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                           input from ASCII to EBCDIC
          OPEN TEACHERS
LOOP      GET    TEACHERS,IREC              Read a single teacher record
          WTO    IREC                       Display the record
          B      LOOP                       Repeat
*
*         EOJ processing
*
ATEND     CLOSE TEACHERS
          RETURN
*
*         Literals, if any, will go here
*
          LTORG
*
*         File definitions
*
TEACHERS DCB     LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
                 DDNAME='TEACHER.DAT'
*
*         Input record definition
*
IREC      DS     CL29                       Teacher record
END     BEGIN
```

```
A:\MIN> teach1a
        732BENSON, E.T.   PHD N5156
        218HINCKLEY, G.B. MBA N5509
        854KIMBALL, S.W.  PHD Y5594
        626YOUNG, B.      MBA Y5664
        574SMITH, J.      MS  Y5320
```

BEGIN... BEGIN El primero BEGIN asigna un nombre a la declaración: algo más que un asterisco en una posición indica una etiqueta, como un

nombre de campo o de Nombre del párrafo. La BEGIN segunda es una macro que guarda los valores de los registros de modo que cuando el programa haya terminado, el funcionamiento sistema será capaz de volver a donde vino. Al igual que con los regs macro, esto es tan común que una macro similar se encuentra en la mayoría de BAL tiendas. Puede llamarse de muchas cosas: lo he visto como BEGIN, BASE13, y GÉNESIS. Para nuestros propósitos, el código es así como se ve aquí.

OI. . . MAESTROS+10,X'08' No voy a explicar las instrucciones en este momento, excepto para decir que este comando debe ser utilizado para todos los archivos. Recordemos que las PC utilizan el ASCII conjunto de caracteres para representar los datos, mientras que los mainframes de IBM (y por lo tanto, PC/370) utilizan el conjunto de caracteres EBCDIC. En consecuencia, todas las entradas deben se convierte de ASCII a EBCDIC medida que se leen, y todas las salidas de se debe convertir de EBCDIC a ASCII como se escriben. Esto la instrucción es el medio por el que dices PC/370 para hacer esto. El archivo MAESTROS nombre (en este caso) se corresponde con el nombre de un archivo en una declaración de DCB (véase más abajo).

Esto es para PC/370 solamente. No hagas esto con mainframe BAL.

OPEN. . . TEACHERS Este comando abre el archivo MAESTROS. Como con toda la programación idiomas, un archivo debe ser abierto antes de que pueda ser utilizado. La macro OPEN PC/370 para no incluye un modo (como entrada o salida), que tendrá que especificar un modo al abrir un archivo con mainframe BAL.

El parámetro de MACRF de la DCB correspondiente indica el modo. indica abierto a la entrada (G = GET), mientras que MACRF = P indica MACRF = G abierto a la producción (P = put). Esto difiere ligeramente de la de mainframe BAL.

Por ahora usted puede estar pensando que PC/370 es muy diferente de BAL mainframe. Realmente no lo es. La mayoría de las diferencias tienen que ver con la la estructura de archivos y macros, que se discute aquí. Convertir de PC/370 para mainframe BAL es realmente trivial!

LOOP. . . GET. . . TEACHERS,IREC Este es el bucle principal del programa, por lo tanto, la etiqueta conseguir maestros, IREC macro lee un registro único de los maestros de archivo en el registro IREC área.

WTO. . . IREC Escribir significa para el operador. Es un macro que, como vimos en el De de la OMC programa, se utiliza para mostrar un mensaje en la pantalla. En este HELLO.MLC caso de que lo están utilizando para mostrar un registro de los profesores de archivo.

Observe que los registros se muestra a doble espacio. Esto se debe a cada registro terminó con CR/LF. La macro de la OMC siempre envía sus propios CR/LF, así, así que en este caso, los registros están a doble espacio. Vamos a arreglar esto en nuestro próximo programa.

B. . . LOOP Esta es una una rama para LOOP. Similar a una declaración de IR en COBOL o BASIC.

ATEND. . . CLOSE. . . TEACHERS es una etiqueta a la que el programa va a poder cuando los profesores archivo está en la final de archivo. El parámetro de la macro EODAD DCB cuenta la ensamblador en la rama al final del archivo. Habiendo llegado al final de la de archivo, cerramos el archivo de la misma manera que en COBOL o BASIC.

RETURN La macro devuelve volver al programa llamando al (por lo general el funcionamiento sistema). Se deshace lo que la macro se BEGIN. Sirve la misma propósito como GoBack en COBOL. Al igual que con la macro BEGIN, esto es así común que una macro similar se encuentra en las tiendas más BAL. Puede muchas cosas se llama: Yo lo he visto como RETURN, EOJ, y el éxodo. Nosotros usan en todos nuestros programas.

26a ⟨*Epilogo* 26a⟩≡ (28 30 32 45 61 67 72 86 97 202) 26b▷

```
        RETURN
```

LTORG Este comando le indica al ensamblador para colocar cualquier literales generado aquí. Los literales se genera cuando se utiliza una constante literal en lugar de la definición de un campo con el mismo valor. Yo siempre este lugar después de la VOLVER declaración. La razón para ello tiene que ver con addressibility; algo que no vamos a discutir en este momento. significa de control de datos de bloque.

26b ⟨*Epilogo* 26a⟩+≡ (28 30 32 45 61 67 72 86 97 202) ◁26a

```
*
*       Literals, if any, will go here
*
        LTORG
```

TEACHERS. . . DCB. . . Hay una DCB para cada archivo en el programa. Dado que este DCB tiene una etiqueta de los profesores, que será el nombre de por la que me refiero a este DCB. La macro DCB tiene muchos parámetros, pero normalmente sólo unos pocos de ellos se utilizan. Son LRECL, RECFM, MACRF, EODAD, y DDNAME.

26c ⟨*TEACHERS DCB, EODAD=ATEND* 26c⟩≡ (28 30 45 61 67 86 97 107 113 187)

```
TEACHERS DCB    LRECL=29,RECFM=F,MACRF=G,EODAD=ATEND,
                DDNAME='TEACHER.DAT'
```

LRECL=29 significa lógico Longitud de registro. El incluye el CR/LF LRECL LRECL discutido anteriormente. (léase como "forma" REC ") significa Record Format.

RECFMT=F Formato de grabación RECFM podría ser F (fijo) o V (variable). RECFM = F indica que todos los registros en el archivo tiene el mismo número de caracteres. RECFM = V indica que el longitud de registro puede variar de un registro a otro. Vamos a discutir cómo BAL trabaja con registros de longitud variable más tarde. Tenga en cuenta que este no es el mismo como registros de longitud variable en un PC, tales como archivos delimitados por comas en BASIC.

MACRF=G significa Macro Formato. Utilizar G (GET) para los archivos de entrada, y P (put) MACRF = G MACRF para los archivos de salida.

EODAD=atEnd Como se mencionó anteriormente, esto indica la etiqueta para que el programa de se debe ir cuando después de intentar leer un registro, un fin de archivo condición fue detectada. Este es el mismo que en el procesamiento de END en COBOL, razón por la cual yo he elegido atEnd de la etiqueta aquí. Este parámetro es no utilizado en los archivos de salida.

DDNAME='TEACHER.DAT' El parámetro DDNAME se utiliza para indicar qué archivo de DOS es para leer de (en el caso de MACRF = G) o creados o sustituido (en el caso de MACRF = P). Esto es similar a la cláusula ASSIGN A de SELECT declaración en COBOL, o el parámetro de nombre de archivo de la instrucción OPEN en la BASIC. (En mainframe BAL, esto se referiría a la DDNAME en el JCL.)

IREC...DS...CL29 significa Definir almacenamiento. Esto indica que el 29 bytes de almacenamiento son DS a reservar para un registro que se conoce como IREC. Esta es la similar a un PIC X (29) en COBOL.

END...BEGIN Este es el final del segmento de programa llamado BEGIN. La BEGIN se refiere a la etiqueta de comienzo de la macro BEGIN discutido anteriormente. Esto siempre ser la última declaración en su programa.

* *

Cuando hablamos de la WTO el comando anterior, señaló que se muestran los registros de a doble espacio, porque cada registro terminó con CR/LF y que la WTO macros siempre envía su propio CR/LF también. Por lo tanto, si queremos que estos registros a aparecer a un solo espacio, necesitamos a la WTO los primeros 27 bytes del registro solamente. Con el fin de hacerlo, deben ser capaces de redefinir nuestro registro de entrada, de modo que podemos referirnos a todo el registro (como el cuando lleguemos a ella), pero sólo muestran una parte de ella. Para ello, tenemos el siguiente código:

```
IREC  DS 0CL29 Teacher record
IDATA DS CL27  Teacher data
ICRLF DS CL2   PC/370 only - CR/LF
```

Hemos sustituido Cl29 con 0CL29. Esto significa que IREC se refiere a la próxima veintinueve bytes, pero que este ámbito se desglosan. Tenga en cuenta que se llaman los veinte primeros siete bytes Idata y bytes veintiocho y veintinueve se llaman ICRLF. En COBOL se vería como:

```
01 IREC.
   05 IDATA PIC X(27).
   05 ICRLF PIC X(2).
```

Luego de la WTO la parte idata solamente. El programa completo, TEACH1B.MLC, y su salida se muestra en la página siguiente.

* *

Hemos introducido el comando de la WTO aquí debido a su simplicidad. En la práctica, la WTO, comando nunca se utilizaría a la lista de los registros en un archivo, como lo hemos hecho aquí. Típicamente el comando se utiliza para el envío de unos pocos, seleccionar los mensajes a la consola, por ejemplo, cómo el programa de avanza (por ejemplo, a partir SORT y COMPLETADO SORT), y para la depuración. Cuándo la creación de un informe, escribir las líneas, o registros, a un archivo. A continuación, puede decidir qué hacer con que archivo. En un PC, podemos usar el comando de tipo DoS "para poder verla, o utilizar el comando COPY para enviar a de la impresora, o importarlo en otro paquete, como un procesador de textos u hoja de cálculo. En el mundo mainframe, usaríamos JCL para asignar el archivo a un dispositivo específico, como una cola de impresión, unidad de cinta, disco, microfichas, etc

Nuestro próximo cambio a este programa quita la y, en cambio, escribe los registros a un archivo. El WTOs los cambios son necesarios los siguientes:

OI...REPORT...10,X'08' Convertir la salida de EBCDIC a ASCII antes de escribir.

OPEN...REPORT Un archivo debe ser abierto antes de que pueda leer o escribir.

PUT...REPORT,IREC Escribir un registro. Nota $LRECL = 29$ en el DCB, la duración del es de 29, y la INFORME DE VENTA, IREC IREC que termina con un CR/LF, que ha sido leído desde el archivo de entrada.

28 ⟨*TEACH1B.MLC, listado de maestros sin LF, version completa* 28⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH1B.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
```

```
*        REMARKS : A card-image list of teacher records.       *
****************************************************************
 ⟨Prologo 23a⟩
BEGIN    BEGIN
         OI     TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                          input from ASCII to EBCDIC
         OPEN TEACHERS
LOOP     GET    TEACHERS,IREC              Read a single teacher record
         WTO    IDATA                      Display the record
         B      LOOP                       Repeat
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
⟨Epilogo 26a⟩
*
*        File definitions
*
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
*
*        Input record definition
*
IREC     DS     0CL29                      Teacher record
IDATA    DS     CL27                       Teacher data
ICRLF    DS     CL2                        PC/370 only - CR/LF
         END    BEGIN
```

```
A:\MIN> teach1b
        732BENSON, E.T.    PHD  N5156
        218HINCKLEY, G.B.  MBA  N5509
        854KIMBALL, S.W.   PHD  Y5594
        626YOUNG, B.       MBA  Y5664
        574SMITH, J.       MS   Y5320
```

REPORT...DCB Cada archivo necesita una DCB.

LRECL=29,RECFM=F Igual que los profesores de archivo.

MACRF=P significa poner, este archivo se abre para la salida.

DDNAME='informe.txt' El nombre del fichero de archivo de disco que contiene los registros de salida es Informe.txt. Si este archivo ya existe, será borrado y un nuevo archivo será creado.

El programa completo, TEACH1C.MLC, y su resultado se muestra en la siguiente página. Observe cómo los DOS tipo se utiliza para mostrar el resultado.

* *

30 ⟨*TEACH1C.MLC, listado de maestros a Archivo, version completa* 30⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH1C.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : A card-image list of teacher records.        *
*****************************************************************
⟨Prologo 23a⟩
BEGIN    BEGIN
         OI     TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                          input from ASCII to EBCDIC
         OI     REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                          output from EBCDIC to ASCII
         OPEN TEACHERS
         OPEN REPORT
LOOP     GET    TEACHERS,IREC              Read a single teacher record
         PUT    REPORT,IREC                Write the record
         B      LOOP                       Repeat
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
⟨Epilogo 26a⟩
*
*        File definitions
*
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT   DCB    LRECL=29,RECFM=F,MACRF=P,
                DDNAME='REPORT.TXT'
*
*        Input record definition
*
IREC     DS     0CL29                      Teacher record
IDATA    DS     CL27                       Teacher data
ICRLF    DS     CL2                        PC/370 only - CR/LF
         END    BEGIN
```

```
A:\MIN> teach1c
A:\MIN> type report.txt
        732BENSON, E.T.     PHD N5156
        218HINCKLEY, G.B.   MBA N5509
        854KIMBALL, S.W.    PHD Y5594
```

```
626YOUNG, B.        MBA Y5664
574SMITH, J.        MS Y5320
```

Un problema con TEACH1C.MLC es que el usuario debe recordar que el informe de salida se encuentra en REPORT.TXT. Vamos a hacer un último cambio de este programa para este capítulo: vamos a agregar tres WTOs como siguiente:

WTO para indicar	colocación de la WTO
Programa ha comenzado la ejecución	Después de la instrucción BEGIN

31a ⟨*TEACH1D.MLC, primer WTO* 31a⟩≡ (32)

```
WTO    'TEACH1D ... Begin execution'
```

WTO para indicar	colocación de la WTO
Archivo de salida está disponible,	después de la declaración CERRAR para ese archivo

31b ⟨*TEACH1D.MLC, segundo WTO* 31b⟩≡ (32)

```
WTO    'TEACH1D ... Teacher list on REPORT.TXT'
```

WTO para indicar	colocación de la WTO
Programa ha terminado normalmente	antes de la instrucción RETURN

31c ⟨*TEACH1D.MLC, tercer WTO* 31c⟩≡ (32)

```
WTO    'TEACH1D ... Normal end of program'
```

El nuevo programa es TEACH1D.MLC: el programa y su producción de seguir.

32 ⟨*TEACH1D.MLC, listado de maestros status a consola, version completa* 32⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH1D.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : A card-image list of teacher records.        *
*****************************************************************
⟨Prologo 23a⟩
BEGIN    BEGIN
⟨TEACH1D.MLC, primer WTO 31a⟩
         OI     TEACHERS+10,X'08'         PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN TEACHERS
         OPEN REPORT
LOOP     GET    TEACHERS,IREC             Read a single teacher record
         PUT    REPORT,IREC               Write the record
         B      LOOP                      Repeat
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
⟨TEACH1D.MLC, segundo WTO 31b⟩
⟨TEACH1D.MLC, tercer WTO 31c⟩
⟨Epilogo 26a⟩
(Remainder of program is unchanged)
```

```
A:\MIN> teach1d
        TEACH1D ... Begin execution
        TEACH1D ... Teacher list on REPORT.TXT
        TEACH1D ... Normal end of program
A:\MIN> type report.txt
        732BENSON, E.T.    PHD N5156
        218HINCKLEY, G.B.  MBA N5509
        854KIMBALL, S.W.   PHD Y5594
        626YOUNG, B.       MBA Y5664
        574SMITH, J.       MS  Y5320
```

1.5. Ejercicios

1. Verdadero o falso.

   a) BAL es un acrónimo para el comienzo de Lenguaje Ensamblador. T F

   b) BAL es un lenguaje de programación de primera generación. T F

   c) Etiquetas, en su caso, debe comenzar en la columna 1. F T

   d) Operaciones previas a su operandos en BAL. F T

   e) Por convenio, la operación se codifica a partir de la columna 10. F T

   f) La extensión de archivo para el código fuente del programador es .MLC. F T

   g) Errores en el código fuente se muestran en el archivo .LST. F T

   h) PC/370 's L370 produce un módulo independiente ejecutable. F T

   i) El CR/LF es una secuencia de caracteres especiales que ocupa cuatro bytes. F T

   j) Algunos editores de PC "trim" espacios en blanco. F T

   k) En PC/370, como ocurre con la mayoría de lenguajes de programación, el modo de archivo se incluye en el F T la instrucción OPEN.

   l) PC utiliza la secuencia de ordenación ASCII. F T

   m) El parámetro LRECL de la DCB debe incluir el CR/LF en su caso. F T

2. Modificar HELLO.MLC para mostrar su nombre y dirección completa en tres líneas.

3. Cree el archivo de los estudiantes a la base de datos Small Town Community College. Llame a su archivo STUDENT.DAT. (ver Tabla 1.1 STUDENT)

Nº	Nombre	Descripción	Inicio	Fin	Long	Formato
1	SID	ID Alumno	1	3	3	ZD
2	SNAME	Nombre	4	18	15	CH
3	SSEX	Sexo	19	19	1	CH
4	SMAR	Estado Civil	20	20	1	CH
5	SCRLF	Sólo pc/370	21	22	2	CR/LF

4. Crear las categorías de archivo de la base de datos Small Town Community College. Llame a su archivo GRADE.DAT. (ver Tabla 1.5 GRADE)

Nº	Nombre	Descripción	Inicio	Fin	Long	Formato
1	SID	ID Alumno	1	3	3	ZD
2	SEM	Semestre	4	6	3	CH
3	CID	ID Curso	7	11	5	CH
4	SECT	Sección	12	12	1	ZD
5	GRADE	Grado	13	13	1	CH
6	GCRLF	Sólo pc/370	14	15	2	CR/LF

5. Crear el archivo del campo de la base de datos Small Town Community College. Llame a su archivo COURSE.DAT. (ver Tabla 1.2 COURSE)

Nº	Nombre	Descripción	Inicio	Fin	Long	Formato
1	CID	ID Curso	1	5	5	CH
2	CDESC	Descripción	6	20	15	CH
3	CHR	Horas	21	21	1	ZD
4	CCRLF	Sólo pc/370	22	23	2	CR/LF

6. Modificar TEACH1D.MLC para producir una lista de los registros en el archivo de los estudiantes, STUDENT.DAT. Ser Asegúrese de cambiar el nombre del programa, los nombres de archivo, y los comentarios como sea necesario para que se apropiado.

7. Modificar TEACH1D.MLC para producir una lista de los registros en el archivo supuesto, COURSE.DAT. Ser Asegúrese de cambiar el nombre del programa, los nombres de archivo, y los comentarios como sea necesario para que se apropiado.

8. Modificar TEACH1D.MLC para producir una lista de los registros en el archivo de notas, GRADE.DAT. Asegúrese para cambiar el nombre del programa, los nombres de archivo, y las observaciones que sea necesario para que se apropiado.

Capítulo 2

Definición y movimiento de Carácteres

Objetivos Al término de este capítulo usted será capaz de:

- Definir los campos alfanuméricos y registros con las instrucciones DS y DC
- Mover campos alfanuméricos utilizando la instrucción de MVC,
- Definir los literales alfanuméricos y los valores equivalentes,
- Mover literales alfanuméricos utilizando el MVI y MVC instrucciones,
- Producir una lista formateada de los registros en un archivo, y
- Producción de líneas informe.

Introducción En el capítulo anterior hemos elaborado un 80/80, o la tarjeta de la imagen, la lista de los registros de la MAESTRO archivo. En este capítulo vamos a continuar con ese ejemplo. En concreto, nos gustaría elaborar el informe con formato de informe y los encabezados de columna. Vamos a empezar por elaborar un rápido y sucio de la lista de registros de este archivo. Por rápido y sucio, queremos decir que, si bien cada campo será en su propia columna, no habrá títulos, no los números de página, etc Nuestro anuncio aparecerá como sigue:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
XXX      XXXXXXXXXXXXXXX        XXXX        X       XXXX
XXX      XXXXXXXXXXXXXXX        XXXX        X       XXXX
XXX      XXXXXXXXXXXXXXX        XXXX        X       XXXX
:       :                       :          :       :
:       :                       :          :       :... Phone
```

```
:      :                    :         :....... Tenured?
:      :                    :.................. Degree
:      :.............................. Nombre
:..................................... ID number
```

Con el fin de hacerlo, debemos ser capaces de

- describir los registros de entrada y salida, y
- mover los campos de entrada a sus respectivos campos de salida.

2.1. Definir y movimiento de Datos de Carácter

La primera instrucción veremos es la instrucción de MVC. MVC significa Movimiento de caracteres. Es utiliza para copiar el contenido de un campo a otro. La instrucción es muy diferente de la instrucción de movimiento (o su equivalente) en otros idiomas. En concreto,

Cuando se utiliza la instrucción MVC, se determina la duración de la medida estrictamente por la longitud del campo de recepción (salvo que se reemplaza), y un máximo de 256 *caracteres se puede mover.*

Para ilustrar esto, vamos a definir tres campos: FLDA es de cinco bytes de largo y contiene 'ABCDE', FLDB es de tres bytes de longitud y contiene "FGH", y FLDC es de cuatro bytes de longitud y contiene 'IJKL. FLDA, FLDB, y FLDC luego ocupar $(5 + 3 + 4) = 12$ bytes contiguos de la memoria. Esto puede ser mostrado como sigue:

A B C D E F G H I J K L

Queremos definir estos campos en el BAL como sigue:

37 ⟨*ex* 37⟩≡ 55a▷

```
FLDA      DC    CL5'ABCDE'
FLDB      DC    CL3'FGH'
FLDC      DC    CL4'IJKL'
```

significa Definir constante. Su propósito es (1) para asignar el espacio, (2) para asignar un nombre a la DC el espacio, (3) para indicar el tipo de datos del espacio se destina a mantener, y (4) para dar a ese espacio un valor inicial.

38 ⟨*ascii* 38⟩≡

```
FLDA       DC    CL5'ABCDE'
:          :      :: :
:          :      :: :... Initial value is ABCDE
:          :      ::           Value is always enclosed in apostrophes
:          :      ::
:          :      ::..... Length of field is five (L5)
:          :      :
:          :      :....... Field is intended to hold character data
:          :
:          :............. Define Constant
:
:..................... Field name is FLDA
```

Para el beneficio del lector que sabe COBOL, COBOL, el equivalente sería la siguiente:

```
05   FLDA      PIC X(5)         VALUE 'ABCDE'.
05   FLDB      PIC X(3)         VALUE 'FGH'.
05   FLDC      PIC X(4)         VALUE 'IJKL'.
```

En COBOL, si el campo de recepción es más corto que el campo de origen, se trunca la mudanza. Para , ejemplo de MOVE FLDA a los resultados de FLDB en un valor de 'ABC' en FLDB, con 'de' truncarse. En el Por otra parte, si el campo de recepción es más largo que el campo de origen, el campo de recepción se rellena con espacios en blanco. Por ejemplo, mueva FLDB a los resultados de FLDA en un valor de 'FGHbb en FLDA (b se utilizan en todo para indicar un blanco. La huelga-a través de se usa para distinguir un espacio en blanco de un b minúscula).

Ahora, vamos a intentar el equivalente BAL de los movimientos de COBOL anterior. Para mover FLDA a FLDB, nos Código:

```
MVC FLDB, FLDA
```

En BAL, la instrucción, o la operación, debe comenzar en algún lugar después de la columna uno. Recordemos que la columna uno es para comentarios y etiquetas. La operación por lo general comienza en la columna de diez en la convención. En menos un espacio en blanco debe separar la instrucción de sus parámetros, o operandos. Los operandos por lo general comienzan en dieciséis columna por convención. Operandos de múltiples se separan por comas, sin espacios en blanco. Opcionalmente, los comentarios pueden ser puestos en el resto de la línea: por lo menos un espacio en blanco debe separar el operando última de los comentarios.

Tenga en cuenta que con la instrucción de MVC (como con la mayoría, pero no todas, las instrucciones de BAL), el primer operando es el campo de recepción. Como se mencionó anteriormente, la duración de la medida está determinada por la longitud de la recepción de campo. En este caso, FLDB es de tres bytes de longitud (CL3 en la instrucción DC), y tres bytes se moverse independientemente de la duración del segundo operando, o el envío de campo.

El resultado es que los tres primeros bytes de "ABCDE" será trasladado a FLDB, el resultado se FLDB contiene 'ABC'. Los mismos doce bytes en memoria ahora debería contener:

```
ABCDEIJKL A B C
```

Vemos que cuando la longitud del campo de recepción es menor o igual a la longitud del envío de campo, MVC BAL funciona igual que MOVE's COBOL.

Vamos a intentar el segundo ejemplo. Para evitar confusiones, se comienza con "fresco" de datos:

```
ABCDEFGHIJKL
```

Para mover FLDB a FLDA, nos código

```
MVC FLDA
```

FLDB En este caso, FLDA es de cinco bytes de longitud (CL5 en la instrucción DC), por lo que cinco bytes se moverá hasta aunque FLDB es de sólo tres bytes de longitud! El resultado dependerá de lo que está en los primeros dos bytes de de memoria inmediatamente después de FLDB. En este caso es 'IJ', los dos primeros bytes de FLDC. Por tanto FLDA contendrá «FGHIJ. Los mismos doce bytes en memoria ahora debería contener:

```
F G H I J K L F G H I J
```

Los resultados probablemente no son lo que usted hubiera esperado! Este concepto de la longitud de un operación que se está determinado por la longitud de un operando (solamente), independientemente de la longitud de la otros, es uno de los conceptos más difícil acostumbrarse a la hora de aprender BAL, por primera vez.

* * * * * * * * * * * * * * * * * * * *

intenta ... Mostrar el resultado de cada uno de los siguientes MVC s. Comenzar con nuevos datos cada vez.

```
Antes de ABCDEFGHIJKL
MVC FLDC, FLDA 1.
MVC FLDC, FLDB 2.
MVC FLDB, FLDC 3.
```

* * * * * * * * * * * * * * * * * * * *

Volvamos ahora al problema en la mano: para producir un rápido y sucio de la lista de los registros de la MAESTRO archivo de acuerdo a las especificaciones. En primer lugar, debemos definir el registro de entrada. Recordemos la diseño de registro para el maestro de archivo:

Campo Nº	Campo Nombre	Descripción	Inicio	Término	Long	Formato
1	TID	ID Maestro	1	3	3	ZD
2	TNAME	Nombre Maestro	4	18	15	CH
3	TDEG	Grado Superior	19	22	4	CH
4	TTEN	Tenured?	23	23	1	S/N
5	TPHONE	Telefono	24	27	4	ZD
6	TCRLF	Sólo PC/370	28	29	2	CR/LF

Tenga en cuenta que he optado por limitar los nombres de campo en todos mis diseños de registro a siete personajes. BAL permite ocho nombres de los campos de caracteres. Los nombres de campos pueden contener nacional (A-Z, @, #, $) o de numérico (0-9) caracteres, y debe comenzar con un carácter nacional. Me he limitado mi nombres de los campos siete caracteres, de modo que puede agregar un carácter más delante, como yo (para los campos de entrada), O (para los campos de producción), o W (para los campos de trabajo).

Como esta es la descripción del registro de entrada, voy a utilizar una I como el primer carácter de cada campo nombre (no porque tenga que hacerlo, sólo porque tiene sentido.) El diseño de registro completo de la siguiente manera:

40 ⟨*ff* 40⟩≡ 41b▷

```
IREC      DS    0CL29          Teacher record
ITID      DS    CL3            Teacher ID nbr
ITNAME    DS    CL15           Teacher name
ITDEG     DS    CL4            Highest degree
ITTEN     DS    CL1            Tenured?
ITPHONE   DS    CL4            Phone nbr
ITCRLF    DS    CL2            PC/370 only - CR/LF
```

Aquí se utiliza la instrucción de DS en lugar de la DC. DS significa Definir almacenamiento. Sirve la misma propósito como DC, salvo que no le asigna un valor inicial. (Si usted está familiarizado con COBOL, un DS es como una definición del campo sin una cláusula de valor.) No es necesario asignar un valor a estos campos. No se hace referencia antes de leer el primer archivo (GET), después de que todos ellos tienen valores.

Los comentarios se utiliza para describir cada campo. Sin embargo, los nombres de campo debe ser tan significativo como posible (con la limitación de ocho caracteres). A menudo incluyen el comienzo y el final de posiciones de la materia en los comentarios. Por ejemplo:

41a ⟨*ejemplo 1* 41a⟩≡

```
IREC      DS    0CL29          1-29    Teacher record
ITID      DS    CL3            1- 3    Teacher ID nbr
ITNAME    DS    CL15           4-18    Teacher name
ITDEG     DS    CL4           19-22    Highest degree
ITTEN     DS    CL1           23-23    Tenured?
ITPHONE   DS    CL4           24-27    Phone nbr
ITCRLF    DS    CL2           28-29    PC/370 only - CR/LF
```

Veamos de nuevo en la 0CL29. En un capítulo anterior nos dice que el cero significa que este campo (IREC) se subdivide. El cero en esta línea es en realidad un multiplicador, lo que significa que son iguales a cero ocurrencias, de 29 bytes, es decir, IREC se define como 29 bytes de longitud, pero no recibe ninguna de almacenamiento de su cuenta. Por el contrario, simplemente se superpone los siguientes campos ya que ocupan el 29 Siguiente bytes.

* *

El uso de multiplicadores de cero es muy común también. Por ejemplo, si quiero un diez campo de un octeto que contiene todos los asteriscos, podría código de cualquiera de los siguientes:

41b ⟨*ff* 40⟩+≡ ◁40 42a▷

```
STARS     DC    CL10'**********'
STARS     DC    C'**********'
STARS     DC    10CL1'*'
STARS     DC    10C'*'
STARS     DC    5CL2'**'
STARS     DC    2CL5'*****'
```

Usted no puede asignar un valor (que utilizan corriente continua) a un campo definido con un cero multiplicador. Además, si intenta para asignar un valor con una DS en lugar de DC, no recibirá un mensaje de error, pero el valor es ignorado. Eso puede ser un error muy difícil de encontrar si no eres consciente de ello!

intenta ¿Cuál de las siguientes definirá veinticuatro campo de un octeto que contiene espacios en blanco?

42a ⟨*ff* 40⟩+≡ ◁41b 47▷

```
4.    BLANKS    DC    CL24' '
5.    BLANKS    DS    4CL6' '
6.    BLANKS    DC    2CL12' '
7.    BLANKS    DC    0CL24' '
8.    BLANKS    DC    24C' '
```

El formato del informe deseado es el siguiente:

```
.       1         2         3         4         5         6       1234567890123456789
XXX     XXXXXXXXXXXXXXX         XXXX       X      XXXX
XXX     XXXXXXXXXXXXXXX         XXXX       X      XXXX
XXX     XXXXXXXXXXXXXXX         XXXX       X      XXXX
:      :                        :         :      :
:      :                        :         :      :... Phone
:      :                        :         :........ Tenured?
:      :                        :.............. Degree
:      :............................... Name
:..................................... ID number
```

Tomar nota del informe es de 60 caracteres de ancho. Pero recuerdo de un debate anterior que, al utilizar PC/370, debemos tener en cuenta el retorno de CR/LF de línea también. Por lo tanto, nuestro informe ser de 62 caracteres de ancho. Así, el parámetro LRECL de la DCB para el Informe se $LRECL = 62$.

42b ⟨*TEACH2A.MLC, linea de salida* 42b⟩≡ (45)

```
OREC      DS      0CL62
OTID      DS      CL3                    Teacher ID nbr
          DC      CL3' '
OTNAME    DS      CL15                   Teacher name
          DC      CL3' '
OTDEG     DS      CL4                    Highest degree
          DC      CL3' '
OTTEN     DS      CL1                    Tenured?
          DC      CL3' '
OTPHONE   DS      CL4                    Phone nbr
          DC      CL21' '
OCRLF     DS      CL2                    PC/370 only - CR/LF
```

Tenga en cuenta que las diferencias entre los campos se han dado los valores iniciales de los espacios en blanco. Podría haber inicializa todos los campos a los espacios en blanco (si se cambia el DSS para países en desarrollo), pero no es necesario, ya que los los campos no se utilizarán hasta que los campos de entrada se han trasladado in A pesar de que cada campo es más de un byte de largo, un espacio en blanco es suficiente. Al definir los datos de caracteres (DC, con el tipo de C), el campo se rellenará con espacios en blanco. Esto no debe confundirse con la instrucción de MVC, que no Pad!

Voy a agregar otra sección para el programa: definiciones de campos diversos. Me gusta cualquier grupo de trabajo campos juntos. Yo a menudo comienzan los nombres de campo con W. sé que mi uso de PC/370 requiere que Puse un CR/LF al final de cada línea de impresión, así que he añadido un campo de trabajo para el programa de este propósito. Este campo se define como:

43a ⟨*TEACH2A.MLC, area de trabajo* 43a⟩≡ (45)

```
*
*           Miscellaneous field definitions
*
WCRLF      DC     X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
```

La X indica que este campo se destina a contener los datos hexadecimales. El 0D hexadecimal corresponde a un retorno de carro EBCDIC, y en hexadecimal 25 corresponde a un EBCDIC salto de línea. Este mismo campo se utiliza en la mayoría de nuestros programas.

Una vez definidos los campos, ahora podemos moverlos. El mover la entrada o campos de trabajo para Los AT los campos de salida correspondientes son:

43b ⟨*TEACH2A.MLC, formateo del reporte* 43b⟩≡ (45)

```
          MVC    OTID,ITID                Move teacher ID Nbr to output
          MVC    OTNAME,ITNAME            Move teacher Name to output
          MVC    OTDEG,ITDEG              Move highest degree to output
          MVC    OTTEN,ITTEN              Move tenure to output
          MVC    OTPHONE,ITPHONE          Move phone nbr to output
          MVC    OCRLF,WCRLF              PC/370 ONLY - end line w/ CR/LF
```

Observe que utilizo WCRLF en lugar de ITCRLF para mover un retorno de carro / avance de línea a OCRLF. Podría han utilizado uno u otro. Yo siempre uso WCRLF. He aquí una razón: tal vez estoy leyendo un archivo que no tiene un CR/LF al final de cada registro. Por ejemplo, tal vez cuando me introducidos los datos, me no se pulse la tecla Intro después de cada 29 bytes de datos. Está bien para hacerlo, y lo hace ahorrar espacio (dos bytes por registro). Pero por lo general, pulse Intro para que pueda ver los datos con más facilidad utilizando DOS ' Comando TYPE, o algo similar. Sin embargo, si no presione Enter, no habría ITCRLF campo, y para producir el informe que tendría que hacer uso de otro campo, tales como WCRLF.

Usted puede ver en la discusión anterior que ITCRLF nunca referencia en este programa. Necesito que los dos bytes de almacenamiento asignado para sostenerlo. Después de todo, está ahí, aunque yo no lo uso. Pero si yo optar por

hacerlo, me pueden dejar fuera el nombre del campo, ya que nunca se utiliza. Por ejemplo, yo podría Código:

44 ⟨*TEACHER IREC, registro de entrada* 44⟩≡ (45 67 86 97 113 187)

```
IREC     DS    0CL29          1-29    Teacher record
ITID     DS    CL3            1- 3    Teacher ID nbr
ITNAME   DS    CL15           4-18    Teacher name
ITDEG    DS    CL4           19-22    Highest degree
ITTEN    DS    CL1           23-23    Tenured?
ITPHONE  DS    CL4           24-27    Phone nbr
ITCRLF   DS    CL2           28-29    PC/370 only - CR/LF
```

Usted verá muchos usos de esta (la omisión de un nombre de campo) cuando hablamos de la creación de líneas de informe en el capítulo siguiente. (Si usted está familiarizado con COBOL, esto es el equivalente a un relleno BAL). En este programa, sin embargo, yo preferiría seguir usando la ITCRLF etiqueta para el bien de la exhaustividad. No me duele nada, y no tiene ningún efecto sobre el tiempo de ejecución de programas.

El programa completo, TEACH2A.MLC, y su producción de seguir.

45 ⟨*TEACH2A.MLC, listado de maestros con formato, version completa* 45⟩≡

```
        PRINT NOGEN
*****************************************************************
*       FILENAME: TEACH2A.MLC                                   *
*       AUTHOR : Bill Qualls                                    *
*       SYSTEM : PC/370 R4.2                                    *
*       REMARKS : A quick-and-dirty list of teachers.           *
*****************************************************************
⟨Prologo 23a⟩
BEGIN   BEGIN
        WTO    'TEACH2A ... Begin execution'
        OI     TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
        OI     REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
        OPEN TEACHERS
        OPEN REPORT
LOOP    GET    TEACHERS,IREC              Read a single teacher record
⟨TEACH2A.MLC, formateo del reporte 43b⟩
        PUT    REPORT,OREC                Write report line
        B      LOOP
*
*       EOJ processing
*
ATEND   CLOSE TEACHERS
        CLOSE REPORT
        WTO    'TEACH2A ... Teacher list on REPORT.TXT'
        WTO    'TEACH2A ... Normal end of program'
⟨Epilogo 26a⟩
*
*       File definitions
*
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT  DCB    LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
⟨TEACH2A.MLC, area de trabajo 43a⟩
*
*       Input record definition
```

```
*
⟨TEACHER IREC, registro de entrada 44⟩
*
*          Output (line) definition
*
⟨TEACH2A.MLC, linea de salida 42b⟩
           END    BEGIN
```

```
A:\MIN> teach2a
TEACH2A ... Begin execution
TEACH2A ... Teacher list on REPORT.TXT
TEACH2A ... Normal end of program
A:\MIN> type report.txt
732     BENSON, E.T.        PHD       N      5156
218     HINCKLEY, G.B.      MBA       N      5509
854     KIMBALL, S.W.       PHD       Y      5594
626     YOUNG, B.           MBA       Y      5664
574     SMITH, J.           MS        Y      5320
```

2.2. Definir y mover literales alfanuméricos

En la sección anterior hemos visto cómo definir y mover los datos de carácter. En concreto, movido los campos de un registro de entrada a los campos correspondientes de un registro de salida. Aquí vamos a buscar la forma de mover las constantes de carácter, o literales. Los siguientes ejemplos ayudan a ilustrar la necesidad de hacerlo:

- Un número de teléfono almacenados como XXXXXXX se va a imprimir como XXX-XXXX. El guión (-) es el
- constante. Un número de seguro social almacena como XXXXXXXXX se va a imprimir como XXX-XX-XXXX. Una vez más, la
- guiones son constantes. Una fecha almacenada como YYMMDD se va a imprimir como MM/DD/19YY. Aquí las diagonales (/) y el 19 son
- constante.

Le mostramos varias maneras de desplazarse literales. Nos concentraremos en los ejemplos primero y tercero. (El segundo ejemplo es muy similar a la primera y se deja como ejercicio.)

* *

Ejemplo # 1 - Un número de teléfono almacenados como se va a imprimir como XXX-XXXX. XXXXXXX

Vamos a definir primero la entrada y salida de la siguiente manera:

47 ⟨ff 40⟩+≡ ◁42a 48a▷

```
IPHONE     DS    0CL7
IPFX       DS    CL3
ILINE      DS    CL4
OPHONE     DS    0CL8
OPFX       DS    CL3
OHYPHEN    DS    CL1
OLINE      DS    CL4
```

Mover el prefijo y la línea no son ningún problema: simplemente utilizar la instrucción MVC como ya hemos discutido:

48a ⟨ff 40⟩+≡ ◁47 48b▷

```
        MVC     OPFX,IPFX
        MVC     OLINE,ILINE
```

Pero, ¿qué acerca de cómo mover el guión? Una solución sería la de definir un campo de trabajo nuevo:

48b ⟨ff 40⟩+≡ ◁48a 53▷

```
WHYPHEN     DC     CL1'-'
```

Podemos código MVC OHYPHEN, WHYPHEN

No hay nada malo con este método, que funciona. Pero hay maneras mejores, tanto en términos de la simplicidad en la codificación, el tiempo de ejecución, y la memoria. Podemos ahorrarnos algo de tiempo en la codificación de utilizando una constante, o literal, en lugar de definir un campo de trabajo con un valor de '-'. Por ejemplo, podemos podría código:

1. MVC OHYPHEN, = CL1'-'
2. MVC OHYPHEN, = C'-'

»» Observe el signo igual es necesario.

Estos dos métodos (con un campo definido vs utilizando un literal) son se ilustra en los próximos dos programas, MOVE2A.MLC y MOVE2B.MLC:

49 ⟨*MOVE2A.MLC, mostrar movimiento de caracteres, version completa* 49⟩≡

```
            PRINT NOGEN
*****************************************************************
*           FILENAME: MOVE2A.MLC                                *
*           AUTHOR : Bill Qualls                                *
*           SYSTEM : PC/370 R4.2                                *
*           REMARKS : Demonstrate character moves.              *
*****************************************************************
            START 0
BEGIN       BEGIN
            WTO    IPHONE
            MVC    OPFX,IPFX
            MVC    OHYPHEN,WHYPHEN
            MVC    OLINE,ILINE
            WTO    OPHONE
            RETURN
*
*           Literals if any will go here
*
            LTORG
*
*         Other field definitions
*
WHYPHEN   DC     CL1'-'
*
IPHONE    DS     0CL7
IPFX      DC     CL3'555'
ILINE     DC     CL4'1212'
*
OPHONE    DS     0CL8
OPFX      DS     CL3
OHYPHEN   DS     CL1
OLINE     DS     CL4
```

```
            END    BEGIN
```

```
A:\MIN> move2a
5551212
555-1212
```

50 ⟨*MOVE2B.MLC, mostrar movimiento de caracteres con =CL1'-', version completa* 50⟩≡

```
            PRINT NOGEN
 ****************************************************************
 *        FILENAME: MOVE2B.MLC                                  *
 *        AUTHOR : Bill Qualls                                  *
 *        SYSTEM : PC/370 R4.2                                  *
 *        REMARKS : Demonstrate character moves.                *
 ****************************************************************
            START 0
 BEGIN      BEGIN
            WTO    IPHONE
            MVC    OPFX,IPFX
            MVC    OHYPHEN,=CL1'-'
            MVC    OLINE,ILINE
            WTO    OPHONE
            RETURN
 *
 *          Literals if any will go here
 *
            LTORG
 *
 *          Other field definitions
 *                                                          Note that the field
 IPHONE     DS     0CL7                                     definition for WHYPHEN
 IPFX       DC     CL3'555'                                 has been removed.
 ILINE      DC     CL4'1212'
 *
 OPHONE     DS     0CL8
 OPFX       DS     CL3
 OHYPHEN    DS     CL1
 OLINE      DS     CL4
            END    BEGIN
```

```
A:\MIN> move2b
5551212
555-1212
```

intenta ... Un número de teléfono con código de área almacena como se va a imprimir como (XXX) XXX-XXXX XXXXXXXXXX

9. Definir el campo de entrada.

10. Definir el campo de salida.

11. Escriba las instrucciones necesarias para mover el campo de entrada al campo de salida. Incluya el guión y paréntesis.

* *

Muchos programadores principiantes están bajo la (equivocada) impresión de que pueden ahorrar memoria mediante literales en lugar de definir las variables con los valores deseados. Esto simplemente no es cierto. Estos dos ejemplos son funcionalmente equivalentes. De hecho, cada vez que un literal es codificada de tal manera, la ensamblador genera una definición de campo, como si había definido uno y darle un nombre. Estos los campos que son generados por el ensamblador se colocan después de la instrucción LTORG (que hemos ya se utiliza, pero desconcertados discutiendo.) Podemos ver esto mediante el examen de la. PRN archivos producidos por la Paso A370. Compruebe los archivos. PRN en la página siguiente y aviso: Tanto las instrucciones de MVC para mover el guión ocupan seis bytes. Esto se puede ver mirando a la

- el código objeto, que se muestra en formato hexadecimal y de la izquierda de la instrucción en el archivo. PRN anuncio.En ambos casos es igual a D200D0ABD0A0. Estos doce dígitos hexadecimales ocupan seis bytes. Esta es la cantidad de memoria requerida por todas las instrucciones de MVC. (Eso es sólo la instrucción, no los datos.) Cuando un literal se utilizó (en MOVE2B.MLC), el ensamblador generado una definición de campo y el lugar que

- después de la LTORG. El resultado es que la misma cantidad de memoria se utiliza para el guión: incluso a pesar de que redujo los requisitos de memoria de un byte cuando nos sacó la definición de WHYPHEN, el ensamblador retrasar un byte para el literal. Por último, la duración total del programa es el mismo: la dirección del primer byte del último campo de

- cada programa es 0000BC.

Tomado de MOVE2A.PRN

```
00007C D202D0A8D0A1       00B8    00B1       29                  MVC             OPFX,IP
000082 D200D0ABD0A0       00BB    00B0       30                  MVC             OHYPHEN
000088 D203D0ACD0A4       00BC    00B4       31                  MVC             OLINE,I
0000AA                                       42   *
0000AA                                       43   *              Literals if any will g
0000AA                                       44   *
0000B0                                       45                  LTORG
0000B0                                       46   *
0000B0                                       47   *              Other field definition
0000B0                                       48   *
```

```
0000B0 60                                49   WHYPHEN     DC          CL1'-'
0000B1                                   50   *
0000B1                                   51   IPHONE      DS          0CL7
0000B1 F5F5F5                            52   IPFX        DC          CL3'555
0000B4 F1F2F1F2                          53   ILINE       DC          CL4'121
0000B8                                   54   *
0000B8                                   55   OPHONE      DS          0CL8
0000B8                                   56   OPFX        DS          CL3
0000BB                                   57   OHYPHEN     DS          CL1
0000BC                                   58   OLINE       DS          CL4
000000                                   59               END         BEGIN
```

Tomado de MOVE2B.PRN

```
00007C D202D0A8D0A1      00B8    00B1    29               MVC         OPFX,IP
000082 D200D0ABD0A0      00BB    00B0    30               MVC         OHYPHEN
000088 D203D0ACD0A4      00BC    00B4    31               MVC         OLINE,I
0000AA                                   42   *
0000AA                                   43   *           Literals if any will g
0000AA                                   44   *
0000B0                                   45               LTORG
0000B0 60                                45                           CL1'-'
0000B1                                   46   *
0000B1                                   47   *           Other field definition
0000B1                                   48   *
0000B1                                   49   IPHONE      DS          0CL7
0000B1 F5F5F5                            50   IPFX        DC          CL3'555
0000B4 F1F2F1F2                          51   ILINE       DC          CL4'121
0000B8                                   52   *
0000B8                                   53   OPHONE      DS          0CL8
0000B8                                   54   OPFX        DS          CL3
0000BB                                   55   OHYPHEN     DS          CL1
0000BC                                   56   OLINE       DS          CL4
000000                                   57               END         BEGIN
```

2.2.1. La Instrucción MVI

Cada vez que un carácter literal de longitud uno se mueve, debe utilizar la instrucción de la MVI en lugar de la instrucción de MVC. La MVI, o paso inmediato, la enseñanza difiere de la instrucción de MVC en de varias maneras:

Con la MVI, el cambio es siempre para un período de uno, independientemente de la duración de la recepción de

- campo. (Recuerde que con MVC, el movimiento es siempre determinada por la longitud de la recepción de de campo). Con la MVI, puede mover

un literal solamente. (Por ejemplo, yo no podía utilizar la MVI para mover la

- régimen de tenencia maestro para el área de salida, a pesar de que el campo se define como un byte en el de longitud.) La instrucción es equivalente MVI MVI OHYPHEN, C »»

Tenga en cuenta que un signo de igual no se utiliza con MVI! Considerando que con la instrucción de MVC, el signo igual era necesario, con la instrucción MVI el signo de igual no está permitido.

El uso de la instrucción de la MVI se ilustra en el siguiente programa, MOVE2C.MLC.

53 ⟨ff 40⟩+≡ ◁48b 56b▷

```
          PRINT NOGEN
*****************************************************************
*         FILENAME: MOVE2C.MLC                                  *
*         AUTHOR : Bill Qualls                                  *
*         SYSTEM : PC/370 R4.2                                  *
*         REMARKS : Demonstrate character moves.                *
*****************************************************************
          START 0
BEGIN     BEGIN
          WTO     IPHONE
          MVC     OPFX,IPFX
          MVI     OHYPHEN,C'-'
          MVC     OLINE,ILINE
          WTO     OPHONE
          RETURN
*
*         Literals if any will go here
*
          LTORG
*
*         Other field definitions
*
IPHONE    DS      0CL7
IPFX      DC      CL3'555'
ILINE     DC      CL4'1212'
*
OPHONE    DS      0CL8
OPFX      DS      CL3
OHYPHEN DS        CL1
OLINE     DS      CL4
          END     BEGIN
```

54 ⟨*out* 54⟩≡

```
A:\MIN>move2c
 5551212
 555-1212
```

Si examinamos el archivo. PRN, se puede comparar a la. PRN archivos producidos por MOVE2A.MLC y MOVE2B.MLC podemos ver que: La instrucción de la MVI para mover el guión sólo ocupa cuatro bytes (frente a los seis bytes para el MVC

- la instrucción.) Esto se puede ver mirando el código objeto, que se muestra en formato hexadecimal ya la izquierda de la instrucción en la lista. PRN. Es igual a 9260D0A2. Estos ocho hex dígitos ocupan cuatro bytes. Esta es la cantidad de memoria requerida por todas las instrucciones de la MVI. El valor de los literales (en este caso '-') es en realidad una parte de la instrucción. Si examinamos la
- de código objeto más de cerca, vemos que el segundo byte ('60 'en '9260D0A2') es el hexadecimal equivalente a un guión. Cuando un MVI se utiliza, el ensamblador no genera un campo que se colocan después de la LTORG. Ya que
- acabamos de ver, no hay necesidad de hacerlo: el valor es en realidad una parte de la instrucción. El programa requiere menos memoria. La dirección del primer byte del último campo en este
- programa es 0000B3. Por último (y esto es un hecho importante, pero no se desprende de la lista), la instrucción de la MVI es
- más rápido de ejecución que es la instrucción de MVC. Claro, no hace ninguna diferencia en estas programas, pero cuando estás de procesamiento cincuenta millones registros, como en un marketing específico la aplicación, la diferencia puede ser importante.

* * * * * * * * * * * * * * * * * * * *

Tomado de MOVE2C.PRN

```
00007C D202D09FD098        00AF    00A8       29                  MVC             OPFX,I
000082 9260D0A2                    00B2       30                  MVI             OHYPHE
000086 D203D0A3D09B        00B3    00AB       31                  MVC             OLINE,
0000A8                             00A8       42   *
0000A8                             00A8       43   *              Literals if any will
0000A8                             00A8       44   *
0000A8                             00A8       45                  LTORG
0000A8                             00A8       46   *
0000A8                             00A8       47   *              Other field definitio
0000A8                             00A8       48   *
```

```
0000A8                          49   IPHONE      DS          0CL7
0000A8 F5F5F5                   50   IPFX        DC          CL3'55
0000AB F1F2F1F2                 51   ILINE       DC          CL4'12
0000AF                          52   *
0000AF                          53   OPHONE      DS          0CL8
0000AF                          54   OPFX        DS          CL3
0000B2                          55   OHYPHEN     DS          CL1
0000B3                          56   OLINE       DS          CL4
000000                          57               END         BEGIN
```

2.2.2. Valores equiparados

Considere las siguientes instrucciones MVI:

55a ⟨*ex* 37⟩+≡ ◁37 55b▷

```
        MVI     ELIGIBLE,C'Y'
        MVI     TAXABLE,C'N'
        MVI     GENDER,C'F'
        MVI     OSLASH,C'/'
```

Cuando se va a mover un Sí o no a un campo, el significado es bastante obvio. En aquellos casos en los que no mover un Si o No, el nombre del campo (como género) a menudo hacen que el significado de la valor inmediato (M o F) obvio. Pero ¿qué pasa con el texto siguiente:

55b ⟨*ex* 37⟩+≡ ◁55a 55c▷

```
        MVI     RACE,C'1'
        MVI     STATUS,C'D'
```

¿Qué hace un RACE = '1 'significa? ¿Cuál es el estado, y lo que hace un valor de 'D' indican? El siguientes son mucho más significativas: donde ...

55c ⟨*ex* 37⟩+≡ ◁55b 55d▷

```
        MVC     RACE,HISPANIC                   HISPANIC DC         CL1'
        MV C    STATUS,DIVORCED                 DIVORCED DC         CL1'
```

Pero nos acaba de decir que debemos utilizar un MVI en lugar de un MVC para mover constantes de la longitud de uno. Tan ¿Cómo podemos utilizar la MVI y aún así obtener las ventajas de un código más significativo? La respuesta es utilizando la instrucción EQU.

El EQU, o igualar, verbo le permite asignar un nombre a un valor inmediato.

A continuación, puede utilizar ese nombre como el segundo operando de una MVI (y otras instrucciones, como veremos más adelante). Para utilizar una EQU con los ejemplos anteriores, nos código: donde ...

55d ⟨*ex* 37⟩+≡ ◁55c 56a▷

```
        MVI     RACE,HISPANIC                   HISPANIC EQU        C'1'
        MVI     STATUS,DIVORCED                 DIVORCED EQU        C'D'
```

Tenga en cuenta que no puede haber ningún operador de longitud en la definición EQU (es decir, usamos C'1 'en lugar de CL1'1 '). MVC.

Valores equiparados no se puede utilizar como operandos de una

Por ejemplo, la siguiente no funcionará: donde ...

56a ⟨*ex* 37⟩+≡ ◁55d 62a▷

```
        MVC    RACE,HISPANIC                    HISPANIC EQU C'1'
        MVC    STATUS,DIVORCED                  DIVORCED EQU C'D'
```

* * * * * * * * * * * * * * * * * * * *

Usted probarlo ... Vuelva a colocar la MVI con MVC en cada uno de los siguientes:

12. MVI tipo, donde C'a 'Type DC CL1' '

13. MVI EOFSW, TRUE donde EOFSW DC CL1 " VERDADERO EQU C'1 ' FALSO EQU C'0 '

14. dónde MVI ERRSW, reconocidos ERRSW DC CL1 " ENCONTRADO EQU C '*'

Teniendo en cuenta SWITCH sustituir la siguiente con los AT MVIS:

```
DC CL1 ' MVC,
Switch vuelta = CL1 '' off MVC, Switch vuelta = CL1'X '
```

en la 15. ... sin ecuaciones

16. ... con las ecuaciones

* * * * * * * * * * * * * * * * * * * *

Nuestro programa anterior, MOVE2C.MLC se ha modificado para utilizar un valor equiparado e incluye: donde ...

56b ⟨*ff* 40⟩+≡ ◁53 57▷

```
  MVI    OHYPHEN,HYPHEN                  HYPHEN      EQU     C'-'
```

El nuevo programa, MOVE2D.MLC, del siguiente modo:

57 ⟨ff 40⟩+≡ ◁56b 61▷

```
          PRINT NOGEN
*****************************************************************
*         FILENAME: MOVE2D.MLC                                  *
*         AUTHOR : Bill Qualls                                  *
*         SYSTEM : PC/370 R4.2                                  *
*         REMARKS : Demonstrate character moves.                *
*****************************************************************
          START 0
BEGIN     BEGIN
          WTO    IPHONE
          MVC    OPFX,IPFX
          MVI    OHYPHEN,HYPHEN
          MVC    OLINE,ILINE
          WTO    OPHONE
          RETURN
*
*         Literals if any will go here
*
          LTORG
*
*         Other field definitions
*
HYPHEN    EQU    C'-'
*
IPHONE    DS      0CL7
IPFX      DC      CL3'555'
ILINE     DC      CL4'1212'
*
OPHONE    DS      0CL8
OPFX      DS      CL3
OHYPHEN   DS      CL1
OLINE     DS      CL4
          END     BEGIN
```

```
A:\MIN> move2d
5551212
555-1212
```

Si examinamos el. PRN archivo y compararlo con el. PRN archivos producidos por MOVE2C.MLC podemos ver que: La instrucción MVI nuevo, utilizando el valor equiparado, creado exactamente el código mismo objeto que la

- a principios de instrucción MVI, es decir, 9260D0A2. El equivalente en hexadecimal de un guión ('60 ') se muestra a la izquierda de la declaración EQU. Esto
- es el mismo '60 'que aparece en la instrucción.

Tomado de MOVE2D.PRN

```
00007C D202D09FD098       00AF      00A8      29                MVC             OPFX,I
000082 9260D0A2                     00B2      30                MVI             OHYPHE
000086 D203D0A3D09B       00B3      00AB      31                MVC             OLINE,
0000A8                                        42   *
0000A8                                        43   *            Literals if any will
0000A8                                        44   *
0000A8                                        45                LTORG
0000A8                                        46   *
0000A8                                        47   *            Other field definitio
0000A8                                        48   *
0000A8                  00000060              49   HYPHEN       EQU             C'-'
0000A8                                        50   *
0000A8                                        51   IPHONE       DS              0CL7
0000A8 F5F5F5                                 52   IPFX         DC              CL3'55
0000AB F1F2F1F2                               53   ILINE        DC              CL4'12
0000AF                                        54   *
0000AF                                        55   OPHONE       DS              0CL8
0000AF                                        56   OPFX         DS              CL3
0000B2                                        57   OHYPHEN      DS              CL1
0000B3                                        58   OLINE        DS              CL4
000000                                        59                END             BEGIN
```

* * * * * * * * * * * * * * * * * * * *

2.3. Longitud explícita y el Desplazamiento

Recordemos que la longitud de una instrucción MVC está determinada por la longitud del campo de recepción. Ya hemos observado lo que puede suceder si el envío y recepción de los campos son de diferentes longitudes. En consecuencia, hemos sido cuidadosos para definir nuestros campos de recepción (en el informe

de salida), que se de la misma longitud que los campos de envío. Pero esto no siempre es posible. Por ejemplo, usted puede tener un programa que tiene dos archivos de entrada (que llamaremos A y B) y un archivo de salida (que llamaremos C). El campo del nombre (con nombre y apellido) se define como 25 bytes en los archivos de A y C, pero como 20 bytes B. A veces en el archivo C obtendrá el nombre de A, mientras que otras veces se recibe el nombre de B.

Mover el nombre de A a C es fácil (MVC CNAME, aName), pero ese no es el caso del traslado del nombre de B a C. No podemos utilizar MVC CNAME, bname porque desde el nombre de C es de cinco octetos ya que el nombre de B, los cinco octetos inmediatamente después del nombre en la letra B se mueve como bien. Lo que necesitamos es una manera de anular la duración de la mudanza. Hay una manera sencilla.

En cualquier MVC, podemos afirmar una longitud explícita. Esta longitud supera la longitud predeterminada, que es la longitud del campo de recepción. Por ejemplo MVC CNAME (20), se moverá bname veinte bytes solamente. El problema con esto es que los octetos 21 a 25 de CNAME se mantendrán sin variación, es decir, si el campo de (o registro) no se ha aclarado, los datos de aName puede ser de sobra en los bytes. Una forma de evitar esto es borrar el campo (o registro) antes de cualquier movimiento que se hace. Por ejemplo MVC CNAME = CL25'b '

En todos nuestros ejemplos, hemos supuesto que los datos que se mueve se va a colocar en el primer byte del campo. Es decir que hay un desplazamiento por defecto, o saltar, de cero. Esto es por lo general, pero no siempre, el caso. Así como nos puede dar una longitud explícita, también podemos dar una explícita desplazamiento. Al hacerlo, no tendrá que talar todo el campo (o registro) como se muestra arriba. Por ejemplo, si estoy en movimiento el nombre de un archivo, puedo código:

```
MVC CNAME, aName
```

como antes, pero si me voy a mudar el nombre del archivo B, voy a código

```
MVC CNAME (20), bname
MVC CNAME 20 (5), = CL5'b '
```

¿Qué pasa si el campo de dirección de la calle se define como 30 bytes en los archivos de A y C, pero como 28 bytes en el fichero B? Una vez más, a veces C obtendrá la dirección de A, mientras que otras veces tomará la dirección de B. Como antes, si me estoy moviendo la dirección de un archivo, puedo código:

```
MVC CSTREET, ASTREET
```

pero si me voy a mudar la dirección del archivo B, voy a código

```
MVC CSTREET (28), BSTREET
MVC CSTREET 28 (2), = CL2'b '
```

El uso excesivo de los literales de los programas más grandes pueden hacer más difícil de mantener. En lugar de han literales de muchos de los type = CL5'b y = CL2'b ', etc, es probablemente mejor para definir un solo campo llamado espacios en blanco o espacios, de una longitud suficiente, como para ser utilizado en lugar de estos literales. Para ejemplo, yo podría código:

```
BLANKS DC CL30'b '
```

y luego el código

```
MVC CNAME,
espacios en blanco MVC CSTREET,
```

espacios en blanco o (cuando el movimiento de datos desde el archivo B)

```
MVC CNAME (20), bname
MVC CNAME 20 (5), espacios en blanco
MVC CSTREET (28), BSTREET
MVC CSTREET 28 (2), espacios en blanco
```

Tenga en cuenta que un desplazamiento explícito puede ser utilizado con MVI también. Sin embargo, una longitud explícita no se puede utilizar con MVI, incluso si la longitud indicada es 1! Por ejemplo, para mover un asterisco a la tercer byte de la basura, puedo código:

```
MVC JUNK 2 (1), = CL1 '*' JUNK
MVI 2, C '*' JUNK MVI 2, STAR (donde STAR EQU C '*')
```

pero no puedo código:

```
MVI JUNK 2 (1), C '*'
```

Recuerde: los valores equiparados no se puede utilizar como operandos de un MVC, y por lo tanto el siguiente no funciona:

MVC JUNK 2 (1), STAR

Cuando se utiliza MVC, el desplazamiento explícito puede ser utilizado tanto en el envío y la recepción de los campos, pero la duración explícita puede ser utilizado en el campo de recepción solamente. Cuando se utiliza la MVI, el desplazamiento explícito puede ser utilizado tanto en el envío y la recepción de los campos, pero la duración explícito no se puede utilizar en absoluto.

Este método de movimiento, con el desplazamiento y la longitud se expresa a veces víctimas de abusos. Unos a los programadores a escribir programas enteros usando este método, a fin de evitar el tiempo y esfuerzo necesarios para el código de diseño de registro. Pero esta es una práctica peligrosa, ya que es muy difícil de depurar. Si ese fuera el único punto negativo, no sería tan malo, porque el programador es sólo herir a sí mismo. Pero estos programas a continuación, obtener pasados a otros programadores que deben mantenimiento. Uso explícito de desplazamiento y la longitud de moderación. Su uso puede ser

apropiado cuando refiriéndose a una parte de un campo, pero rara vez sería apropiado cuando se refiere a una parte de un registro. TEACH2A.MLC

El siguiente programa, TEACH2B.MLC, es funcionalmente equivalente al programa que se muestra anterior, pero utiliza el desplazamiento explícito y longitud solamente. ¿Qué preferirías mantener?

61 ⟨*ff* 40⟩+≡ ◁57 64▷

```
          PRINT NOGEN
 ****************************************************************
 *         FILENAME: TEACH2B.MLC                                *
 *         AUTHOR : Bill Qualls                                 *
 *         SYSTEM : PC/370 R4.2                                 *
 *         REMARKS : Don't do it this way!!!                    *
 ****************************************************************
⟨Prologo 23a⟩
BEGIN     BEGIN
          WTO     'TEACH2B ... Begin execution'
          OI      TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                            input from ASCII to EBCDIC
          OI      REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                            output from EBCDIC to ASCII
          OPEN TEACHERS
          OPEN REPORT
LOOP      GET     TEACHERS,IREC
          MVC     OREC(3),IREC
          MVC     OREC+6(15),IREC+3
          MVC     OREC+24(4),IREC+18
          MVC     OREC+31(1),IREC+22
          MVC     OREC+35(4),IREC+23
          MVC     OREC+60(2),=X'0D25'
          PUT     REPORT,OREC                Write report line
          B       LOOP
*
*         EOJ processing
*
ATEND     CLOSE TEACHERS
          CLOSE REPORT
          WTO     'TEACH2B ... Teacher list on REPORT.TXT'
          WTO     'TEACH2B ... Normal end of program'
⟨Epilogo 26a⟩
*
*         File definitions
*
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT     DCB     LRECL=62,RECFM=F,MACRF=P,
                   DDNAME='REPORT.TXT'
```

```
*
*          Field definitions
*
IREC       DS      CL29                        Teacher record
OREC       DC      CL62' '                     Report line
          END     BEGIN
```

* * * * * * * * * * * * * * * * * * * *

Ejemplo # 2 - Un número de seguro social almacena como XXXXXXXXX se va a imprimir como XXX-XX-XXXX.

Este ejemplo es similar al ejemplo 1 anterior, y se deja como ejercicio.

* * * * * * * * * * * * * * * * * * * *

Ejemplo # 3 - A la fecha almacenada como YYMMDD se va a imprimir como MM/DD/19YY.

Definición de todos los campos, y el uso de los AT sólo, sin literales.

62a ⟨*ex* 37⟩+≡ ◁56a 62b▷

```
                MVC     OMM,IMM
                MVC     OSLASH1,SLASH
                MVC     ODD,IDD
                MVC     OSLASH2,SLASH
                MVC     O19,NINETEEN
                MVC     OYY,IYY
```

donde

62b ⟨*ex* 37⟩+≡ ◁62a 62c▷

```
IDATE       DS      0CL6
IYY         DS      CL2
IMM         DS      CL2
IDD         DS      CL2
ODATE       DS      0CL10
OMM         DS      CL2
OSLASH1     DS      CL1
ODD         DS      CL2
OSLASH2     DS      CL1
O19         DS      CL2
OYY         DS      CL2
SLASH       DC      CL1'/'
NINETEEN    DC      CL2'19'
```

Definición de todos los campos, pero el uso de los AT y MVIS con literales.

62c ⟨*ex* 37⟩+≡ ◁62b 63a▷

```
                MVC     OMM,IMM
                MVI     OSLASH1,C'/'
                MVC     ODD,IDD
                MVI     OSLASH2,C'/'
                MVC     O19,=CL2'19'
                MVC     OYY,IYY
```

en todos los campos se definen como antes. Usando el desplazamiento explícito y longitud solamente, y los valores equiparados utilizando.

63a ⟨*ex 37*⟩+≡ ◁62c 63b▷

```
        MVC     ODATE(2),IDATE+2
        MVI     ODATE+2,SLASH
        MVC     ODATE+3(2),IDATE+4
        MVI     ODATE+5,SLASH
        MVC     ODATE+6(2),=CL2'19'
        MVC     ODATE+8(2),IDATE
```

donde

63b ⟨*ex 37*⟩+≡ ◁63a 65b▷

```
IDATE   DS      CL6
ODATE   DS      CL10
SLASH   EQU     C'/'
```

Usted probarlo ... Dado A CL8 DC y B DC CL6 '. El campo de A contiene una fecha almacenada en DD-MM-AA formato. Mover A a B, tal que B contiene la fecha en formato YYMMDD ...

17. ... la definición de todos los campos y el uso de los AT solamente.

18. ... longitud explícita el uso y el desplazamiento solamente.

19. Se intenta repetir el ejercicio 11 utilizando la longitud explícita y el desplazamiento solamente. Informe

2.4. Agregar y encabezados de columna

La anterior lista de los registros en el archivo maestro fue mencionado como una "rápida y sucia" de la lista: nos dijo que por "rápido y sucio" nos referimos a un informe sin títulos, los números de página, etc Nuestro propósito aquí es añadir informe y los encabezados de columna a la lista de los profesores.

No vamos a hacer números de página en este momento. Los números de página puede parecer una cosa fácil para usted si usted ya saben otra lengua, pero ese no es el caso en ensamblador. Lo hacen, después de todo, requieren aritmética (es decir, añadir 1 a la página de venta libre) y la impresión de los resultados de la aritmética no es trivial, cosa en BAL. Nuestra nueva lista aparecerá como sigue:

```
1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                              LIST OF TEACHERS
ID#            Name              Degr      Ten Phone
---      ---------------         ----      --- -----
XXX      XXXXXXXXXXXXXXX         XXXX        X      XXXX
```

```
XXX     XXXXXXXXXXXXXXX         XXXX      X      XXXX
XXX     XXXXXXXXXXXXXXX         XXXX      X      XXXX
```

Usaremos DS y DC, para definir las líneas. Nota:

- Existen cuatro líneas de partida (la segunda es los espacios en blanco),
- Cada uno debe ser definida como 62 bytes desde LRECL = 62, y
- Cada línea de partida debe terminar con CR/LF (PC/370 solamente).

Hay muchas maneras de las partidas en formato. La elección es simplemente una cuestión de preferencia personal. I mostrará varios de los métodos más comunes, así como el método que prefiera. Me centraré en la tercera línea de partida: las técnicas utilizadas se aplicará a las otras líneas también. Por lo general mi etiqueta partidas como HD1, HD2, etc Una técnica para la definición de lo anterior (el tercero) la partida es el siguiente:

64 ⟨*ff* 40⟩+≡ ◁61 65a▷

```
HD3         DS      0CL62
            DC      CL3'ID#'                    3
            DC      CL8' '                      8
            DC      CL4'Name'                   4
            DC      CL9' '                      9
            DC      CL4'Degr'                   4
            DC      CL2' '                      2
            DC      CL3'Ten'                    3
            DC      CL2' '                      2
            DC      CL5'Phone'                  5
            DC      CL20' '                    20
            DC      XL2'0D25'                   2
                                                62
```

Nota:

- Más de un DC se utiliza para definir el título, por lo que HD3 es un DS (no DC) con un multiplicador de cero,
- La suma de las longitudes de campo es de 62, y de Posiciones
- 61-62 de la partida se define como un CR/LF.

De vez en cuando se utiliza el método anterior, pero cuando lo hago, prefiero incluir las posiciones de impresión como comentarios. Esto mejora la mantenibilidad del programa.

65a ⟨*ff* 40⟩+≡ ◁64 67▷

```
HD3         DS      0CL62
            DC      CL3'ID#'            1- 3     3
            DC      CL8' '              4-11     8
            DC      CL4'Name'          12-15     4
            DC      CL9' '             16-24     9
            DC      CL4'Degr'          25-28     4
            DC      CL2' '             29-30     2
            DC      CL3'Ten'           31-33     3
            DC      CL2' '             34-35     2
            DC      CL5'Phone'         36-40     5
            DC      CL20' '            41-60    20
            DC      XL2'0D25'          61-62     2
                                               62
```

A pesar de que el espacio entre columnas es más de un byte de ancho, un espacio en blanco se utilizó como el valor de cada uno de los países en desarrollo. Esto es así porque, como hemos dicho anteriormente, al definir los datos de carácter (DC, con el tipo C), el campo se rellenará con espacios en blanco, es decir, los siguientes son equivalentes: es equivalente a

65b ⟨*ex* 37⟩+≡ ◁63b 65c▷

```
DC      CL8'bbbbbbbb '                  DC          CL8'b '
```

Muchos programadores va un paso más allá, al darse cuenta que, como este es el caso, se podría incluir que la longitud de la longitud del campo anterior, con lo que la omisión de los países en desarrollo en blanco por completo:

65c ⟨*ex* 37⟩+≡ ◁65b 66a▷

```
HD3         DS      0CL62
            DC      CL11'ID#'            1-11     11
            DC      CL13'Name'          12-24     13
            DC      CL6'Degr'           25-30      6
            DC      CL5'Ten'            31-35      5
            DC      CL25'Phone'         36-60     25
            DC      XL2'0D25'           61-62      2
                                                62
```

Yo prefiero a romper mis partidas en bloques de 40 caracteres. Este tamaño se encajan fácilmente en una sola línea, y dado que la mayoría de los gráficos de diseño de impresión tienen líneas de la cuadrícula cada 10 caracteres, es un simple la tarea de transcribir los títulos de la tabla de diseño de impresión de código ensamblador real.

Por ejemplo:

66a ⟨*ex* 37⟩+≡ ◁65c 66b▷

```
HD3         DS    0CL62
            DC    CL40'ID#              Name                    Degr     Ten    Phone'
            DC    CL20' '
            DC    XL2'0D25'
```

La ventaja de esta técnica es más evidente cuando hay varias líneas de partida, cada uno de más de 80 caracteres de ancho. (La mayoría de los informes de mainframe están diseñados para ser 132 caracteres de ancho, exclusivo de los el carácter de control de carro. Esto se discute en el apéndice.)

Las definiciones completas partidas son las siguientes:

66b ⟨*ex* 37⟩+≡ ◁66a 72▷

```
*
*           Headings definitions
*
HD1         DS    0CL62
            DC    CL40'                  LIST OF TEACHERS                        '
            DC    CL20' '
            DC    XL2'0D25'
HD2         DS    0CL62
            DC    CL60' '
            DC    XL2'0D25'
HD3         DS    0CL62
            DC    CL40'ID#              Name                    Degr     Ten    Phone'
            DC    CL20' '
            DC    XL2'0D25'
HD4         DS    0CL62
            DC    CL40'--       --------               --     --    ---'
            DC    CL20' '
            DC    XL2'0D25'
```

Después de que el archivo de salida (informe) es abierto, y antes de leer los registros, cada una de las líneas de partida que está escrito. El comando PUT se utiliza para este propósito. El nuevo programa es TEACH2C.MLC: la programa y sus resultados los siguientes:

67 ⟨*ff* 40⟩+≡ ◁65a 101▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH2C.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : List of teachers, with headings.             *
*****************************************************************
⟨Prologo 23a⟩
BEGIN    BEGIN
         WTO    'TEACH2C ... Begin execution'
         OI     TEACHERS+10,X'08'         PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN TEACHERS
         OPEN REPORT
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
LOOP     GET    TEACHERS,IREC             Read a single teacher record
         MVC    OTID,ITID                 Move teacher ID Nbr to output
         MVC    OTNAME,ITNAME             Move teacher Name to output
         MVC    OTDEG,ITDEG               Move highest degree to output
         MVC    OTTEN,ITTEN               Move tenure to output
         MVC    OTPHONE,ITPHONE           Move phone nbr to output
         MVC    OCRLF,WCRLF               PC/370 ONLY - end line w/ CR/LF
         PUT    REPORT,OREC               Write report line
         B      LOOP
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         WTO    'TEACH2C ... Teacher list on REPORT.TXT'
         WTO    'TEACH2C ... Normal end of program'
⟨Epilogo 26a⟩
*
*        File definitions
*
```

⟨ *TEACHERS DCB, EODAD=ATEND* 26c ⟩

```
REPORT   DCB    LRECL=62,RECFM=F,MACRF=P,
                         DDNAME='REPORT.TXT'
*
*        Miscellaneous field definitions
*
WCRLF    DC     X'0D25'                    PC/370 ONLY - EBCDIC CR/LF
*
*         Input record definition
*
```

⟨ *TEACHER IREC, registro de entrada* 44 ⟩

```
*
*         Output (line) definition
*
OREC     DS      0CL62
OTID     DS      CL3                        Teacher ID nbr
         DC      CL3' '
OTNAME   DS      CL15                       Teacher name
         DC      CL3' '
OTDEG    DS      CL4                        Highest degree
         DC      CL3' '
OTTEN    DS      CL1                        Tenured?
         DC      CL3' '
OTPHONE  DS      CL4                        Phone nbr
         DC      CL21' '
OCRLF    DS      CL2                        PC/370 only - CR/LF
*
*         Headings definitions
*
HD1      DS      0CL62
         DC      CL40'                   LIST OF TEACHERS                    '
         DC      CL20' '
         DC      XL2'0D25'
HD2      DS      0CL62
         DC      CL60' '
         DC      XL2'0D25'
HD3      DS      0CL62
         DC      CL40'ID#              Name                    Degr Ten Phone'
         DC      CL20' '
         DC      XL2'0D25'
HD4      DS      0CL62
         DC      CL40'--      --------               -- -- ---'
         DC      CL20' '
         DC      XL2'0D25'
         END     BEGIN
```

```
A:\MIN> teach2c
        TEACH2C ... Begin execution
        TEACH2C ... Teacher list on REPORT.TXT
        TEACH2C ... Normal end of program
A:\MIN> type report.txt
        LIST OF TEACHERS
        ID#          Name            Degr     Ten      Phone
        ---    ---------------       ----     ---      -----
        732    BENSON, E.T.          PHD        N      5156
        218    HINCKLEY, G.B.        MBA        N      5509
        854    KIMBALL, S.W.         PHD        Y      5594
        626    YOUNG, B.             MBA        Y      5664
        574    SMITH, J.             MS         Y      5320
```

2.5. Ejercicios

1. 1. Verdadero o falso.

   a) La instrucción MVC FLDA, FLDB se moverá (copia) FLDA a FLDB. F T

   b) El número de personajes se movían con un solo MVC está determinada por la longitud de F T del campo de envío de menos que se sustituya.

   c) Un máximo de 256 caracteres se puede mover con un único MVC. F T

   d) Al utilizar un MVC, si el campo de recepción es más corto que el campo de envío, el F T personajes de la extrema derecha se truncan (no se mueven).

   e) Al utilizar un MVC, si el campo de recepción es más largo que el campo de envío, el F T bytes adicionales (derecha) del campo de recepción se rellenan con espacios en blanco.

   f) DÓLARES DC 5CL4'$$$' asignará un total de 15 bytes. F T

   g) El MVC y las instrucciones de la MVI ocupar la misma cantidad de memoria cuando cada uno es F T mover un campo de un octeto.

   h) El desplazamiento explícito puede ser utilizado con MVC y la MVI. F T

   i) Longitud explícita se puede especificar en una MVI si dicha duración es de uno. F T

   j) El uso de valores equiparados puede hacer MVIS más legible. F T

   k) Dado SIETE DC CL1'7 ', una MVI debe utilizarse en lugar de un MVC para mover SIETE F T a otro campo de un byte.

l) Los siguientes son equivalentes: X DC CL5'ABC 'y X DC CL5'ABCbb' F T

m) Todos los DSS y países en desarrollo requieren nombres de campo (etiquetas). F T

2. 2. Teniendo en cuenta las siguientes definiciones de los campos adyacentes, determinar el resultado de cada uno de los siguientes instrucciones. Comience con nuevos datos para cada pregunta.

```
Un CL5'JKLMN DC '
B DC CL4'PQR '
C DC CL3'ST '
X DC CL1'W '
Y EQU C'Z '
```

a) (B) MVC, A B serán los siguientes:

b) MVC A, B, A será:

c) MVC B, C, B, serán los siguientes:

d) Un MVC 2 (2), C A será: (e) MVC A 3 (2), B 3 A será:

e) MVC C (1), X C será:

f) MVC C (1), C = c'x 'serán los siguientes:

g) MVI B 2, B C'Y 'serán los siguientes:

h) MVI B +2, Y B serán:

3. Mostrar cómo se compara podría ser utilizado para hacer las siguientes declaraciones más significativas:

```
a. MVI SEXO, B. C'M '
MVI DAYOFWK, Domingo C'1 '
MVI SEXO,
MVI C'F 'DAYOFWK, de lunes C'2' :
MVI DAYOFWK, sábado C'7 '
```

4. Utilizando los ejemplos dados en este capítulo (MOVE2A.MLC, MOVE2B.MLC, MOVE2C.MLC, y MOVE2D.MLC), escribir y ejecutar cuatro programas de BAL para demostrar los diferentes métodos de por el que se podría completar los movimientos descritos en el Ejemplo # 2: Una seguridad social número almacenado como XXXXXXXXX se va a imprimir como XXX-XX-XXXX. Mire el resultado. PRN archivos. Discutir las diferencias y similitudes en términos de facilidad de codificación, el tiempo de ejecución, uso de memoria, y el mantenimiento.

5. En el marketing directo (también conocido como correo no deseado) una tarea común es la fusión / purga. En el combinar / purgar, varias listas (como las listas de correo de las casas de diferente orden y la lista de

correo los corredores) se combinan, y los duplicados se quitan con el fin de reducir al mínimo la impresión y gastos de envío costes. Con el fin de identificar los duplicados, un código de partido suele ser generado. Hay muchos formas de generar un código de partido. Por ejemplo, el código de partido puede consistir en la cremallera código, las primeras letras, tercero y cuarto del apellido, y la primera (hasta seis) números consecutivos de la dirección de la calle.

Así pues, dada:

```
KATHY NEGRO
618 S ANZA
PASADENA CA 91106
```

El código de partido sería: 91106BAC000618

Tenga en cuenta los siguientes nombres daría el código de un mismo partido: Kathi NEGRO, CATHY NEGRO, K. NEGRO, J. y C. NEGRO.

Su tarea es la siguiente: El registro de entrada es de 80 bytes de longitud, con el código postal en las posiciones 1-5, el último nombre en las posiciones 21-32, y la dirección de la calle en las posiciones 33-62. Todos los demás las posiciones no han sido utilizados en este ejemplo. El código de partido es de 14 bytes de longitud como se ilustra arriba. Usted está en el código del DS y MVC necesario para mover el código postal para las posiciones 1-5 del Código del partido, y para mover las posiciones 1, 3 y 4 del apellido a las posiciones 6, 7 y 8 (respectivamente) del Código del partido. (Tenemos que hacer nada con la dirección de la calle en este ejemplo.)

Challenge - Código de la DS y MVC de manera que el nombre se puede mover con dos niños más vulnerables.

6. Producir una lista formateada de los registros en el archivo del estudiante. Cada línea debe ser 33 bytes de largo, exclusiva de la CR/LF. El informe deberá incluir el informe y los encabezados de columna. El formato deseado es el siguiente:

```
         1         2         3
12345678901234567890123456789012
STUDENT MASTER LIST
ID#     Student Name           Sex       Mar
---    ---------------         ---       ---
XXX    XXXXXXXXXXXXXXX           X        X
XXX    XXXXXXXXXXXXXXX           X        X
XXX    XXXXXXXXXXXXXXX           X        X
```

7. Producir una lista formateada de los registros en el archivo de grado. Mostrar el semestre, el ID del curso, identificación de estudiante, y la calificación obtenida solamente, en ese orden. Cada línea debe ser de 50

bytes de longitud, exclusiva de la CR/LF. El informe deberá incluir el informe y los encabezados de columna. La deseada formato es el siguiente:

```
          1         2         3         4         5
12345678901234567890123456789012345678901234567890
GRADE MASTER LIST
(Confidential)       Course       Student
Sem    Number           ID#         Grade
---    ------       -------         -----
XXX    XXXXX            XXX             X
XXX    XXXXX            XXX             X
XXX    XXXXX            XXX             X
```

8. Producir una lista formateada de los registros en el archivo del curso. Mostrar el ID del curso y curso única descripción. Mostrar el número de curso que el departamento (las dos primeras posiciones de la ID del curso) y el número del curso (tercera, cuarta y quinta posición de la ID del curso) se separados por un espacio único. Cada línea debe ser de 30 bytes de longitud, exclusiva de la CR/LF. El el informe deberá incluir el informe y los encabezados de columna. El formato deseado es el siguiente:

```
          1         2         3
123456789012345678901234567890
COURSE MASTER LIST
Course      Description
------    ---------------
XX XXX    XXXXXXXXXXXXXX
XX XXX    XXXXXXXXXXXXXX
XX XXX    XXXXXXXXXXXXXX
```

9. El siguiente programa hace (excesiva) el uso de longitud explícita y el desplazamiento de elaborar una lista de campos seleccionados de la oferta de cursos de archivo. Ejecutar el programa como es el determinar la salida. A continuación, realice los cambios necesarios para limpiar el código, mientras que todavía producir los mismos resultados. Su solución no debe utilizar la longitud explícita y desplazamiento a todos!

72 ⟨*ex* 37⟩+≡ ◁66b

```
          PRINT NOGEN
*****************************************************************
*         FILENAME: OFFER2A.MLC                                 *
*         AUTHOR :                                              *
*         SYSTEM : PC/370 R4.2                                  *
*         REMARKS : A quick-and-dirty list of offerings.        *
*****************************************************************
⟨Prologo 23a⟩
BEGIN     BEGIN
```

```
          OI      OFFER+10,X'08'
          OI      REPORT+10,X'08'
          OPEN OFFER
          OPEN REPORT
LOOP      GET     OFFER,IREC
          MVC     OREC(2),IREC+3
          MVI     OREC+2,C'-'
          MVC     OREC+3(3),IREC+5
          MVI     OREC+6,C'-'
          MVC     OREC+7(1),IREC+8
          MVC     OREC+11(1),IREC
          MVC     OREC+12(3),=C'*19'
          MVC     OREC+15(2),IREC+1
          MVC     OREC+20(4),IREC+12
          MVC     OREC+27(3),IREC+9
          MVC     OREC+31(2),=C'**'
          MVC     OREC+33(2),=X'0D25'
          PUT     REPORT,OREC
          B       LOOP
ATEND     CLOSE OFFER
          CLOSE REPORT
⟨Epilogo 26a⟩
OFFER     DCB     LRECL=18,RECFM=F,MACRF=G,EODAD=ATEND,
                  DDNAME='OFFER.DAT'
REPORT    DCB     LRECL=35,RECFM=F,MACRF=P,
                  DDNAME='OFFER.TXT'
IREC      DS      CL18
OREC      DC      CL35' '
          END     BEGIN
```

Capítulo 3

IFS en BAL: Comparación de carácter FIELDS

Objectivos Al término de este capítulo será capaz de:

- Determinar el código de estado resultante de la comparación alfanuméricos utilizando el CLC y las instrucciones de la CLI,
- Explicar cómo la rama sobre la condición de la instrucción se utiliza para alterar la secuencia de transformación, según el código de condición,
- los mnemónicos lista ampliada de la rama en la instrucción condición
- Usar la CLC y las instrucciones de la CLI para aplicar la lógica en un diagrama de flujo determinado o pseudocódigo, y
- Escribir un programa de extracción.

Introducción En los capítulos anteriores hemos visto varias formas de elaborar una lista de los registros en el archivo MAESTRO. En cada caso hemos enumerado todo el archivo. Veamos ahora cómo la lista de registros seleccionados. Estos programas se conocen comúnmente como "extracto" programas.

Nuestro próximo programa será elaborar una lista de profesores titulares. El nuevo informe aparecerá como sigue:

```
1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                    LIST OF TENURED INSTRUCTORS
ID#           Name              PhD?         Phone
---    ---------------          ----       --------
XXX    XXXXXXXXXXXXXXX           X         517-XXXX
XXX    XXXXXXXXXXXXXXX           X         517-XXXX
```

```
XXX   XXXXXXXXXXXXXXX        X        517-XXXX
                              :        :
                              :        :..... All campus phones
                              :                  begin with '517-'
                              :
                              :.......... 'Y' if highest degree
                                                is PhD, 'N' otherwise
```

Con el fin de extraer los registros, tenemos que aprender el equivalente a la BAL SI verbo como en muchos otros idiomas. En este programa, vamos a necesitar el SI para dos propósitos:

- Para determinar si un profesor es titular y, por consiguiente, incluirse en el informe, y
- Para determinar si el instructor tiene un doctorado, y para imprimir una Y o una N en consecuencia.

3.1. Compara en BAL: el código de condición

Cuando dos campos son comparados en el BAL, un código de condición se pone. *Eso es todo lo que sucede: un código de condición se pone.* El código de estado es una zona especial de cuatro bits de lado en la CPU para este propósito. (Otra instrucción, Subdivisión de la condición, se utiliza para indicar qué medidas deben tomarse con base en el código de estado resultante. Esta instrucción se explica más adelante en este capítulo.)

Hay tres condiciones posibles como resultado de una operación de comparación:

- La primera operando es igual al segundo operando,
- El primer operando es menor que el segundo operando, o
- El primer operando es mayor que el segundo operando.

El código de condición de cada una de las condiciones antes mencionadas:

	1er bit	2ndo bit	3er bit	4to bit	Decimal	Resultado
A = B	1	0	0	0	8	equal
A < B	0	1	0	0	4	low
A > B	0	0	1	0	2	high

Tenga en cuenta que en el BAL, no hay equivalente a una prueba para la igualdad, o una prueba para el superior, etc No es sólo una comparación, que establece un código de condición. Le corresponde entonces a probar el código de condición y tienen la rama del programa, o "Ir a", la ubicación deseada en su programa en consecuencia.

3.2. Compara en BAL: La Instrucción CLC

El formato general para comparar es un personaje CLC op1, op2

significa Comparar carácter lógico. Funciona de la misma manera como la instrucción de MVC en CLC que la duración de la comparación (el número de bytes en comparación) está determinada por la longitud del primer operando (en este caso, OP1), independientemente de la duración del segundo operando. Un máximo de 256 bytes se puede comparar con un único CLC.

Vamos a tratar algunos compara, y determinar el código de estado resultante. Nos referiremos a los siguientes datos a través de nuestra discusión de IFS en BAL:

76a ⟨*Field A* 76a⟩≡

```
AREC      DS    0CL28                        1-28
ANAME     DS    0CL16                        1-16
AFIRST    DS    CL8                          1- 8
ALAST     DS    CL8                          9-16
AZIP      DS    CL5                         17-21
AAGE      DS    CL2                         22-23
          DS    CL2                         24-25
ASEX      DS    CL1                         26-26
ACRLF     DS    XL2     PC/370 Only         27-28
```

76b ⟨*Field B* 76b⟩≡

```
BREC      DS    0CL28                        1-28
BNAME     DS    0CL18                        1-18
BFIRST    DS    CL9                          1- 9
BLAST     DS    CL9                         10-18
BZIP      DS    CL5                         19-23
BAGE      DS    CL2                         24-25
BSEX      DS    CL1                         26-26
BCRLF     DS    XL2     PC/370 Only         27-28
```

76c ⟨*Field C* 76c⟩≡

```
CREC      DS    0CL28                        1-28
CNAME     DS    0CL14                        1-14
CFIRST    DS    CL7                          1- 7
CLAST     DS    CL7                          8-14
CZIP      DS    CL9                         15-23
CAGE      DS    CL2                         24-25
CSEX      DS    CL1                         26-26
CCRLF     DS    XL2     PC/370 Only         27-28
```

También se hará referencia a las siguientes áreas de trabajo:

77 ⟨*Field W* 77⟩≡

```
BLANKS     DC      CL20' '
WNAME      DS      0CL20
WFIRST     DS      CL10
WLAST      DS      CL10
WZIP       DS      CL5
WAGE       DS      CL2
WSWITCH    DS      CL1
```

The Data

```
                  1         2
         12345678901234567890123456
  File A CHERYL HAVLIK 6018333 F
  File B APRIL      HAVLIK      6055430F
  File   KEVEN FOOTE 60183025828M
```

* * * * * * * * * * * * * * * * * * * *

1. CLC AFIRST,BFIRST

   Desde AFIRST se define como CL8, un total de ocho octetos serán comparados, aunque BFIRST se define como CL9: el último byte de BFIRST no está incluido en la comparación.

   ```
   CLC      AFIRST,BFIRST
   :          :          |C|H|E|R|Y|L|b |b|.........
   :
   :
   :          |A|P|R|I|L|b |b|b|................:
   ```

   La comparación se realiza de izquierda a derecha, comparando el primer byte de cada campo, luego el segundo, el tercero, etc, hasta que todos los bytes se han comparado o hasta que se encontró una diferencia. En este caso, el primer byte es diferente (C vs A) para la comparación termina después de un solo byte. Y puesto que C es mayor que (viene después) A, el código de estado resultante es 0010, o 2, o alta.

2. CLC AZIP, BZIP

   Desde AZIP se define como CL5, un total de cinco bytes serán comparados. El hecho de que BZIP también se define como CL5 es irrelevante.

   ```
   CLC      AZIP,BZIP
                              :       :
   |6|0|1|8|3|...............:             :
   |6|0|5|5|4|...................:
   ```

La primera diferencia que se produce en el tercer byte. Desde el 1 es inferior a 5, AZIP es inferior a BZIP y el código de estado resultante es 0100, o 4, o baja.

3. CLC CSEX, BSEX

   Desde CSEX se define como CL1, la primera (y única) de bytes de cada campo será comparado. El hecho de que BSEX también se define como CL1 es irrelevante.

```
 CLC      CSEX,BSEX
                              :       :
|M|.......................:
                                      :
                                      :
|F|............................:
```

   Como M es mayor que (viene después de F), el código de estado resultante es 0010, o 2 o alto.

4. CLC AZIP, CZIP

   Desde AZIP se define como CL5, un total de cinco octetos serán comparados, aunque CZIP se define como CL9: los cuatro bytes de la derecha CZIP no están incluidos en la comparación.

```
CLC      AZIP,CZIP
                              :       :
|6|0|1|8|3|...............:       :
                                    :
|6|0|1|8|3|...................:
```

   Los cinco octetos son iguales. El código de estado resultante es de 1000, o 8, o igual.

5. CLC CZIP, AZIP

   A primera vista, puede parecer que la respuesta debe ser la misma que para el número 4, pero ese no es el caso. Desde CZIP se define como CL9, un total de nueve octetos serán comparados, aunque AZIP se define como CL5. Los nueve octetos comparación será el contenido de AZIP además de los próximos cuatro bytes en el registro, es decir, la edad (dos bytes) y un relleno (dos bytes).

```
CLC      CZIP,AZIP
                                                  :       :
|6|0|1|8|3|0|2|5|8|.......:              :
                                                          :
|6|0|1|8|3|3|3|b |b|............:
```

La primera diferencia que se produce con el sexto byte. Dado que 0 es menor que 3, el código de estado resultante es 0100, o 4, o baja.

* * * * * * * * * * * * * * * * * * * *

You Try It... Determine el código de estado resultante para la siguiente comparación:

```
1.    CLC    ALAST,BLAST
2.    CLC    BLAST,ALAST
```

* * * * * * * * * * * * * * * * * * * *

Con la CVX, como con MVC, el desplazamiento explícito y / o longitud puede ser utilizado para anular el desplazamiento por defecto (cero) y longitud (la longitud del campo), respectivamente. El desplazamiento explícito puede ser utilizado en cualquiera de los operandos, mientras que la longitud explícita puede ser utilizado en el primer operando solamente. Por ejemplo, para comparar octetos 3-5 de FLDA con bytes 2-4 de FLDB, código:

```
          CLC    FLDA+2(3),FLDB+1
```

Los literales se puede utilizar con la CVX. Por ejemplo, los siguientes son válidas y funcionalmente equivalentes:

```
          CLC    OPTION,=CL3'YES'
          CLC    OPTION,=C'YES'
          CLC    OPTION,YES
               where
YES        DC    CL3'YES'
```

3.3. Compares in BAL: The CLI Instruction

El CLI, o comparar lógica inmediata, la instrucción es la CVX como MVI es MVC: lo utilizan para comparar un único byte en un literal. Por ejemplo, para comprobar si el primer byte de un campo es un asterisco, el código:

```
          CLI    FLDA,C'*'
```

Al igual que con MVI:

- El desplazamiento explícito puede ser utilizado,
- Longitud de explícito no puede ser utilizado,
- No existe ningún signo igual en el literal, y
- Los valores equiparados puede ayudar a mejorar la legibilidad y facilidad de mantenimiento.

You Try It... Determine el código de estado resultante para la siguiente comparación:

```
3.    CLI     ASEX,C'M'
4.    CLI     CZIP,ZERO
        where     ZERO EQU C'0'
```

* * * * * * * * * * * * * * * * * * * *

3.4. Compares in BAL:The BC Instruction

Una vez que el código de condición se ha fijado, se utiliza la Subdivisión de Estado en la instrucción para indicar a dónde ir basado en el valor del código de condición. La Subdivisión de la condición, o antes de Cristo, la instrucción tiene dos operandos: el valor de prueba, o la máscara, para el código de condición, y la etiqueta para que el programa debe rama. Por ejemplo:

```
              CLC     A,B
              BC      8,EQUAL
              BC      4,ALOW
              BC      2,AHIGH
              BC      7,NOTEQUAL                Note: 7 = 15 - 8
              BC      11,ANOTLOW                     11 = 15 - 4
              BC      13,ANOTHIGH                    13 = 15 - 2
                        :
EQUAL         EQU     *                       :
AHIGH         EQU     *                       :                    etc.
```

La instrucción antes de Cristo funciona como sigue: después de comparar, el código de condición se pone. La máscara en la instrucción se compara con el código de condición. Si alguna "en" bits en la máscara tiene un correspondiente "en" bits en el código de condición, entonces se toma la rama. Por ejemplo:

```
10 00         Condition code                          8: A = B
```

Tenga en cuenta que el nuevo uso de la ecuación en el ejemplo de arriba! Un asterisco, cuando se utiliza como operando en una instrucción, se refiere a la dirección de esa instrucción. Aquí, la ecuación se utiliza para equiparar una etiqueta con una dirección en el programa. Esa es una forma elegante de decir la etiqueta EQU * es una forma conveniente de definir un párrafo en el BAL. Se utilizará en todos los ejemplos y los programas a partir de ahora.

```
BE          Branch   on  A  Equal B
       BH          Branch   on  A  High
       BL          Branch   on  A  Low
       BNE         Branch   on  A  Not Equal B
```

```
BNH        Branch   on  A  Not High
BNL        Branch   on  A  Not Low
```

Así, por ejemplo, las instrucciones previas antes de Cristo podría reescribirse como:

```
CLC     A,B
 BE      EQUAL
 BH      AHIGH
 BL      ALOW
 BNE     NOTEQUAL
 BNH     ANOTHIGH
 BNL     ANOTLOW
```

Una BC importante sigue siendo: BC 15 es un salto incondicional. Es simplemente su mnemónico B.

Todas estas ramas son los mismos que GOTO en otros idiomas. Si le han enseñado a utilizar la programación estructurada o, más concretamente, "go-to-menos" de programación, esto puede no sienta bien con usted. Sin embargo, acostumbrarse a él! No se puede hacer ir a menos de programación en el BAL. (Hay algunas macros de uso común que permiten a un IF-ENDIF construir, pero son una extensión de la LBA y no se discutirán aquí.)

* * * * * * * * * * * * * * * * * * * *

You Try It... 5. Will BC 8,SKIP branch to SKIP if the condition code is 8?

6. Will BC 12,SKIP branch to SKIP if the condition code is 2?
7. Will BC 15,SKIP branch to SKIP if the condition code is 8?
8. Will BE SKIP branch to SKIP if the condition code is 8?
9. Will B SKIP branch to SKIP if the condition code is 4?

* * * * * * * * * * * * * * * * * * * *

3.5. Compares in BAL: Sample IFs

normalmente no es el primer idioma que aprende una persona. Lo que sigue es una manera fácil de aprender a hacer BAL FI en BAL que se aprovecha de los conocimientos de programación existentes.

Para aprender a hacer las federaciones en BAL, usted debe pensar muy bajo nivel. BAL es, después de todo, un nivel bajo (de segunda generación) de idiomas. Así que vamos a dar un paso atrás en el tiempo. Vamos a suponer que estamos codificando en una forma muy primitiva de BASIC. Esta versión de BASIC es totalmente desestructurado y tiene las siguientes restricciones:

1. Sólo hay una línea por cada declaración. (En las versiones más recientes de BASIC de dos puntos se pueden utilizar para poner varias instrucciones BASIC en la misma línea. Este no será el caso en BASIC nuestro "viejo".)

2. Hay un verbo SI, SI, pero este no permite que las condiciones de compuestos (AND y OR).

3. Lo único que puede hacer como resultado de un SI es un GOTO. (Esto era realmente el caso de algunas versiones anteriores de BASIC!)

4. Además, en nuestra "vieja base", lo único que puede GOTO es una REM (comentario) Declaración, específicamente, puede GOTO a una THEN REM, REM ELSE, ENDIF REM, REM y, o real o solamente.

Eso es lo que es la codificación de IFS en BAL! Considere los siguientes ejemplos. (Si usted no sabe BASIC pero sé otra lengua, debe ser capaz de comprender los ejemplos. No te cuelgan en la sintaxis básica, tales como signos de dólar para indicar las variables de cadena, etc Es el concepto de FI de programación en un lenguaje de bajo nivel que más nos interesa aquí.)

Una nota sobre los diagramas de flujo contenida en este documento Si está familiarizado con la programación estructurada, entonces usted sabe que todos los problemas de programación se pueden resolver mediante tres simples constructos: secuencia, selección (IF-THEN-ELSE) y la iteración (DoWhile). Todos los diagramas de flujo debe ser construido de manera que se pueden desglosar en combinaciones de estos tres constructos solamente. Los diagramas de flujo que se ve aquí no se ajustan a estas construcciones. La razón es que estas construcciones (en particular la selección y la iteración) generalmente permiten las condiciones de compuestos (AND y OR). Desde BAL no permite que estas condiciones compuesto, selecciones múltiples deben ser utilizados, es decir, un simple DoWhile en algunos idiomas se requieren CLC múltiples en BAL. Estos diagramas de flujo podrían haber sido escritas para que fueran plenamente estructurado, pero para ello sería necesario repetir el código para el entonces y / o acciones ELSE. Los diagramas de flujo aquí se han elaborado para reflejar el código, ya que generalmente se escribe en el mundo real.

Example #1 - IF..THEN

```
Pseudocode
          IF AZIP is equal to BZIP
              move AZIP to WZIP
          ENDIF
Flowchart
                AZIP =                                          Y
```

"Old BASIC " Solution

```
10 IF AZIP$ <> BZIP$ THEN GOTO 30
20 LET WZIP$ = AZIP$
30 REM ENDIF
```

Solution BAL

```
                          CLC  AZIP,BZIP
                           BNE  SKIP
                           MVC  WZIP,AZIP
                 SKIP      EQU  *
```

Example #2 - IF..THEN..ELSE

```
          Pseudocode
                 IF AZIP is less than BZIP
                     move AZIP to WZIP
                 ELSE
                     move BZIP to WZIP
                 ENDIF
          Flowchart
                              AZIP <                          N               Y
```

"Old BASIC " Solution

```
10  IF AZIP$ >= BZIP$ THEN GOTO 40
20  LET WZIP$ = AZIP$
30  GOTO 60
40  REM ELSE
50  LET WZIP$ = BZIP$
60  REM ENDIF
```

Solution BAL

```
                          CLC  AZIP,BZIP
                           BNL  USEB
                           MVC  WZIP,AZIP
                           B    SKIP
                 USEB      EQU  *
                           MVC  WZIP,BZIP
                 SKIP      EQU  *
```

Example #3 - IF..AND..THEN..ELSE

```
          Pseudocode
                 IF (BAGE is equal to CAGE) AND (BSEX = CSEX)
                     move BSEX to WSEX
                     move 'Y' to WSWITCH
                 ELSE
                     move CSEX to WSEX
                     move 'N' to WSWITCH
                 ENDIF
          Flowchart
                               BAGE =                     N               Y
```

"Old BASIC " Solution

```
10  IF BAGE$ <> CAGE$ THEN GOTO 60
20  IF BSEX$ <> CSEX$ THEN GOTO 60
30  LET WSEX$ = BSEX$
40  LET WSWITCH$ = "Y"
50  GOTO 90
60  REM ELSE
70  LET WSEX$ = CSEX$
80  LET WSWITCH$ = "N"
90  REM ENDIF
```

Solution BAL

```
                          CLC     BAGE,CAGE
                           BNE     NOPE
                           CLC     BSEX,CSEX
                           BNE     NOPE
                           MVC     WSEX,BSEX
                           MVI     WSWITCH,C'Y'
                           B       DONE
               NOPE        EQU     *
                           MVC     WSEX,CSEX
                           MVI     WSWITCH,C'N'
               DONE        EQU     *
Example #4 - IF..OR..THEN
      Pseudocode
              IF (AAGE greater than 30) OR (ASEX is male)
                  move AAGE to WAGE
                  move ASEX to WSEX
              ENDIF
      Flowchart
                   AAGE >                              Y                    30?
```

"Old BASIC " Solution

```
10  IF AAGE$ > "30" THEN GOTO 40
20  IF ASEX$ = "M" THEN GOTO 40
30  GOTO 70
40  REM THEN
50  LET WAGE$ = AAGE$
60  LET WSEX$ = ASEX$
70  REM ENDIF
```

Solution BAL

```
                        CLC     AAGE,=CL2'30'
                         BH      YES
                         CLI     ASEX,C'M'
                         BE      YES
                         B       MORE
                YES      EQU     *
                         MVC     WAGE,AAGE
                         MVC     WSEX,ASEX
                MORE     EQU     *
```

* * * * * * * * * * * * * * * * * * * *

Ejemplo completa Suponga que desea determinar si un campo contiene un segundo nombre inicial solamente y, en caso afirmativo, para mover un período posterior a la inicial. Suponga que la definición del campo y el valor siguientes:

```
MNAME    DS      CL12
       |P|b |b|b|b|b|b|b|b|b|b|b|
```

We will assume that there is an initial (only) if

- the first byte is not blank, and
- the remaining bytes are blank.

One solution is as follows:Vamos a suponer que existe una inicial (sólo) si

- El primer byte no está en blanco, y
- Los bytes restantes están en blanco.

Una solución es la siguiente:

```
                CLI     MNAME,C' '
                 BE      NOINIT
                 CLC     MNAME+1(11),=CL11' '
                 BNE     NOINIT
                 MVI     MNAME+1,C'.'
          NOINIT EQU     *
```

Observe el uso de EQU * para definir una etiqueta para que el programa puede rama, o "ir a", basado en el código de estado resultante de CLI o CLC.

3.6. Sample Program

Recordemos que el problema de programación presentado en el comienzo de este capítulo fue:

- Para determinar si un profesor es titular y, por consiguiente, incluirse en el informe, y
- Para determinar si el instructor tiene un doctorado, y para imprimir una S o una N en consecuencia.

El nuevo programa es TEACH3A.MLC: el programa y sus resultados los siguientes:

86 ⟨*cap 3* 86⟩≡ 97▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH3A.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce a list of tenured instructors.       *
*****************************************************************
⟨Prologo 23a⟩
BEGIN    BEGIN
         WTO    'TEACH3A ... Begin execution'
         OI     TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                           input from ASCII to EBCDIC
         OI     REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                           output from EBCDIC to ASCII
         OPEN   TEACHERS
         OPEN   REPORT
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
LOOP     EQU    *
         GET    TEACHERS,IREC              Read a single teacher record
         CLI    ITTEN,C'Y'                 Is teacher tenured?
         BNE    LOOP                       No, then skip this record
         MVC    OTID,ITID                  Move teacher ID Nbr to output
         MVC    OTNAME,ITNAME              Move teacher Name to output
         CLC    ITDEG,=CL4'PHD'            Highest degree = PhD?
         BE     YESPHD                     .. Yes, branch
         MVI    OPHD,C'N'                  .. No, Show PhD = 'N'
         B      OTHERS                     .. Branch around YES logic
YESPHD   EQU    *                          Highest degree is PhD, so...
         MVI    OPHD,C'Y'                  Show PhD = 'Y'
OTHERS   EQU    *                          Continue moving other fields...
         MVC    O517,=CL4'517-'            All phone nbrs begin w/ '517-'
         MVC    OTPHONE,ITPHONE            Move phone nbr to output
         MVC    OCRLF,WCRLF                PC/370 ONLY - end line w/ CR/LF
         PUT    REPORT,OREC                Write report line
```

```
         B      LOOP
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         WTO    'TEACH3A ... Teacher list on REPORT.TXT'
         WTO    'TEACH3A ... Normal end of program'
⟨Epilogo 26a⟩
*
*        File definitions
*
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT   DCB    LRECL=62,RECFM=F,MACRF=P,
                DDNAME='REPORT.TXT'
*
*        Miscellaneous field definitions
*
WCRLF    DC     X'0D25'                    PC/370 ONLY - EBCDIC CR/LF
*
*        Input record definition
*
⟨TEACHER IREC, registro de entrada 44⟩
*
*        Output (line) definition
*
OREC      DS      0CL62
OTID      DS      CL3                       Teacher ID nbr
          DC      CL3' '
OTNAME    DS      CL15                      Teacher name
          DC      CL4' '
OPHD      DS      CL1                       PhD? (Y/N)
          DC      CL5' '
OPHONE    DS      0CL8                      Phone nbr
O517      DS      CL4
OTPHONE   DS      CL4                       Phone nbr
          DC      CL21' '
OCRLF     DS      CL2                       PC/370 only - CR/LF
*
*         Headings definitions
*
HD1       DS      0CL62
          DC      CL40'          LIST OF TENURED INSTRUCTORS                    '
          DC      CL20' '
          DC      XL2'0D25'
HD2       DS      0CL62
```

```
          DC      CL60' '
          DC      XL2'0D25'
HD3       DS      0CL62
          DC      CL40'ID#              Name                    PhD?         Phone  '
          DC      CL20' '
          DC      XL2'0D25'
HD4       DS      0CL62
          DC      CL40'--       --------              --       ---- '
          DC      CL20' '
          DC      XL2'0D25'
          END     BEGIN
```

```
A:\MIN> teach3a
        TEACH3A ... Begin execution
        TEACH3A ... Teacher list on REPORT.TXT
        TEACH3A ... Normal end of program
A:\MIN> type report.txt
              LIST OF TENURED INSTRUCTORS
        ID#          Name             PhD?        Phone
        ---    ---------------       ----       --------
        854    KIMBALL, S.W.           Y        517-5594
        626    YOUNG, B.               N        517-5664
        574    SMITH, J.               N        517-5320
```

3.7. Exercicios

1. Verdadero o falso.

   a) _____ El código de estado es un área especial de la CPU que consta de ocho bits.

   b) El equivalente de BAL de si el género es masculino THEN GO TO domale requiere dos instrucciones TF.

   c) BC 15, PASE es un salto incondicional. F T

   d) El 8 en BC 8, PASE es también conocida como la máscara. F T

   e) Hasta 256 bytes se puede comparar con una instrucción CLI único. F T

   f) Valores equiparados se utilizan comúnmente con CLI. F T

   g) CLI INICIAL, = C ',' y CLC inicial (1), C ',' cada uno comparar el primer byte de la TF inicial a un blanco.

   h) ELSE EQU * se puede utilizar varias veces en un programa. F T

   i) BE es un ejemplo de un extendido mnemotécnico. F T

   j) BC 2, XXX y XXX BL son equivalentes. F T

k) A fin de aplicar y de las condiciones o, la CLC y las instrucciones de la CLI permite TF comparaciones múltiples en una única instrucción.

l) El desplazamiento explícito puede ser utilizado con CLI. F T

m) Longitud explícita puede ser utilizado con CLI. F T

2. Teniendo en cuenta los campos de entrada siguientes:

```
NAZIP9     DS     0CL9        ZIP CODE 9 DIGITS
NAZIP5     DS     CL5         ..ZIP CODE FIRST 5
NAZIP4     DS     CL4         ..ZIP CODE "PLUS 4"
```

Escriba el código de BAL necesario formatear el código postal de la impresión según las definiciones de los campos siguientes:

```
PRZIP9     DS     0CL10      FORMATTED 9-DIGIT ZIP CODE
PRZIP5     DS     CL5        .. ZIP CODE FIRST 5
PRDASH     DS     CL1        .. HYPHEN IF "PLUS 4" EXISTS
PRZIP4     DS     CL4        .. "PLUS 4" IF IT EXISTS
```

(Un "plus 4" existe si no es en blanco y no todos los ceros).

3. Teniendo en cuenta:

```
FLDA       DS     CL3
FLDB       DS     CL3
FLDC       DS     CL3
MAX        DS     CL3
```

Escriba el código de BAL para mover el máximo de FLDA, FLDB, y FLDC a MAX.

4. Habida cuenta de las definiciones de los campos siguientes ...

```
NACITY     DS      CL12         56-67       CITY
NASTATE    DS      CL2          68-69       STATE
NAPHONE    DS      0CL10        70-79       PHONE...
NAAREA     DS      CL3          70-72       ...AREA CODE
NAEXCH     DS      CL3          73-75       ...EXCHANGE
NALINE     DS      CL4          76-79       ...LINE
NACODE     DS      CL1          80-80       TRANS CODE (A/C/D)
WSWITCH    DS      CL1          (Y)ES OR (N)O
```

De las federaciones siguiente diagrama de flujo, y escribir la correspondiente "viejo BASIC" y el código de BAL.

```
a.      IF state is Illinois
                move yes to switch
            ENDIF
b.      IF state is Utah or California
                move yes to switch
            ELSE
                move no to switch
            ENDIF
c.      IF trans code is other than 'A', 'C', 'D'
                move yes to switch
            ELSE
                move no to switch
            ENDIF
                                                          Careful! Watch
d.      IF area code = "312"
                                                          the length on the
                                                          CLC for the city!
            ENDIF
```

5. Utilizando las definiciones de campo en las páginas 2-3 de este capítulo, las federaciones siguiente diagrama de flujo y escribir la correspondiente "viejo BASIC" y el código de BAL:

```
a.      IF (ANAME is blank) AND (CNAME is blank)
                move all File B fields to corresponding work area fields
            ELSE
                IF (AAGE is greater than or equal to CAGE)
                    move all File A fields to corresponding work area fields
                ELSE
                    move all File C fields to corresponding work area fields
                ENDIF
            ENDIF
b.      IF ((ASEX is male) AND (AAGE is 18 or over))
            OR ((ASEX is female) AND (AAGE is 21 or over))
                move 'Y' to WSWITCH
            ELSE
                move 'N' to WSWITCH
            ENDIF
```

6. El siguiente es un extracto de un catálogo de la escuela: Los individuos en una o más de las siguientes categorías se colocan en el asesoramiento académico:

   - los estudiantes que tienen una terminación Promedio general por debajo de 2,40 en cualquiera de los siguientes del tercer curso.
   - los estudiantes con más de tres abandonos.

- los estudiantes con más de tres ciclos repetidos

Habida cuenta de las definiciones de los campos siguientes, determinar si un estudiante debe ser colocado en el asesoramiento académico:

```
GPA         DS      CL3
Example: 2.97 = '297'
#COMP       DS      CL2     Number of courses completed
#WD         DS      CL1     Number of withdrawls
#REP        DS      CL1     Number of repeated courses
ADVISE      DS      CL1     Academic advisory (Y/N)
```

7. El siguiente es un extracto de un catálogo de la escuela: Reglamentos de ayuda financiera del gobierno federal requieren que los receptores de ayuda financiera para avanzar progresivamente hacia el logro de sus grados. A fin de mantener la elegibilidad para ayuda financiera federal, los estudiantes deben hacer progresos adicionales de acuerdo a la tabla siguiente: Años transcurridos número mínimo de cursos desde el inicial efectuada con el paso de grado

```
enrollment              MBA             MHRM                MPM
```

Habida cuenta de las definiciones de los campos siguientes, determinar si un estudiante es elegible para ayuda financiera federal (FFA):

```
YEARS       DS      CL1     Years elapsed
#COMP       DS      CL2     Number of courses completed
DEGREE      DS      CL4     Degree sought
FFA         DS      CL1     Fed Fin Aid eligibility (Y/N)
```

8. Teniendo en cuenta las siguientes definiciones de los campos adyacentes:

```
BOY     DC      CL7'WILLIAM'
GIRL1   DC      CL4'CORA'
GIRL2   DC      CL6'HANNAH'
GIRL3   DC      CL4'Emma'
MID     DC      CL3'KAY'
BLANKS  DC      CL5' '
MISC    DS      CL12
```

Para cada uno de los siguientes CLC y CLI, determinar qué caracteres son de hecho que se comparan, y luego determinar el código de estado resultante (8, 4, o 2). Si alguna de las compara produciría un error del compilador, a continuación, que se lo indiquen.

```
a.      CLC    GIRL1,GIRL2
b.      CLC    BOY+2(1),BOY+3
c.      CLC    BOY+2(2),BOY+3
d.      CLC    GIRL2+2(1),GIRL2+3(1)
e.      CLC    BOY+4(4),=C'LIAM'
f.      CLC    GIRL1+3(2),C'AH'
g.      CLC    BOY+6(1),GIRL3+2
h.      CLC    GIRL3+3(1),GIRL2
i.      CLI    GIRL1,=C'A'
j.      CLI    GIRL3+3,C'A'
```

Hannah's middle name is Kay. Write the code necessary to move " HANNAH " to MISC KAY

utilizando los campos de arriba solamente.

9. Dibuja el diagrama de flujo para el código de BAL siguiente. Suponga que todos los campos son CL1.

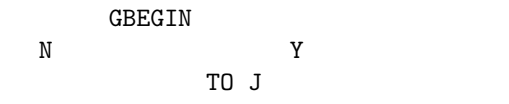
```
SBEGIN     EQU    *
                          MVC    T,S
                          CLC    T,U
                          BNL    SEND
                          CLC    T,V
                          BNH    SEND
                          CLC    T,W
                          BNE    SKIP
                          MVC    V,W
                          B      SEND
                  SKIP    EQU    *
                          MVC    V,T
                  SEND    EQU    *
```

10. Dibuja el diagrama de flujo para el código de BAL siguiente. Suponga que todos los campos son CL1.

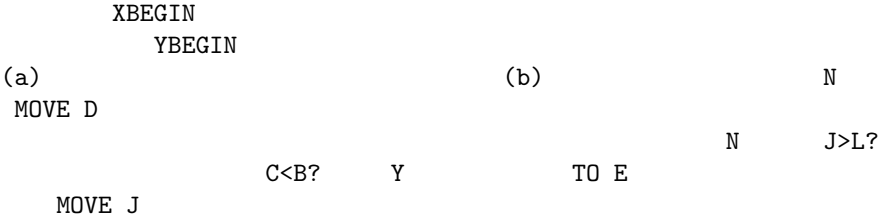
```
                  ZBEGIN      EQU      *
                                 MVC      S,R
                                 CLC      S,T
                                 BH       ZSKIP2
                                 CLC      S,U
                                 BNL      ZSKIP1
                                 MVC      T,R
                                 B        ZEND
                     ZSKIP1      EQU      *
                                 MVC      R,T
                                 B        ZEND
                     ZSKIP2      EQU      *
```

```
                    MVC     S,U
                    CLC     S,T
                    BE      ZSKIP3
                    MVC     S,Q
                    CLC     Q,U
                    BNL     ZSKIP4
                    MVC     T,S
                    B       ZSKIP4
ZSKIP3              EQU     *
                    MVC     T,Q
ZSKIP4              EQU     *
                    MVC     R,T
ZEND                EQU     *
```

11. Escriba el BASIC "viejo" y el BAL para el siguiente diagrama de flujo. Nota: G y H se define como CL4, mientras que J y K se define como CL1. Identificar las partes del diagrama de flujo y el código que nunca se ejecutará.



12. Escriba el BASIC "viejo" y el BAL de los diagramas de flujo siguiente. Suponga que todos los campos se definen como CL1.



13. Write the BAL code for the following pseudocode.

```
(a)       IF (X > Y)
              IF (X > Z)
                   MOVE Z TO W
              ELSE
                   IF (X = Z)
                        MOVE W TO Z
                   ENDIF
                   MOVE Y TO W
              ENDIF
          ENDIF
```

```
                IF (A = B) AND (C != D)
        (b)                     MOVE X TO Y
                ELSE
                     IF (A > B) OR (C <= D)
                           MOVE W TO Y
                     ENDIF
                     MOVE X TO Z
                ENDIF
```

14. Los chicos Chi Psi han pedido una lista de solo mujeres estudiantes. Escriba el programa para producir una tal lista del archivo de estudiante. El informe debe aparecer como sigue:

```
.        1         2         3
12345678901234567890123456789012345
                      ELIGIBLE PARTIES
                     FOR OUR NEXT PARTY
ID#      Student Name          Sex       Mar
---     ---------------       ---       ---
XXX     XXXXXXXXXXXXXXX         X        X
XXX     XXXXXXXXXXXXXXX         X        X
XXX     XXXXXXXXXXXXXXX         X        X
```

15. Escribir un programa que producirá una lista formateada de los cursos que están a menos de 3 horas semestrales de crédito. El informe debe aparecer como sigue:

```
.        1         2         3         4
123456789012345678901234567890123456789012
      COURSES WITH LESS THAN 3 HOURS CREDIT
Course         Description            Hours
------       ---------------          -----
XXXXX        XXXXXXXXXXXXXXX             X
XXXXX        XXXXXXXXXXXXXXX             X
XXXXX        XXXXXXXXXXXXXXX             X
```

16. El departamento de Inglés ha solicitado una lista de todos los grados para las clases de Inglés para el año escolar 1992-93 (F92 y W93 semestres solamente.) Escribir un programa para elaborar dicha lista con el siguiente formato:

```
.        1         2         3         4         5
12345678901234567890123456789012345678901234567890
  ENGLISH GRADES FOR 1992-93 SCHOOL YEAR
(Confidential)      Course       Student
Sem    Number          ID#         Grade
---    ------      -------         -----
```

```
XXX     XXXXX           XXX             X
XXX     XXXXX           XXX             X
XXX     XXXXX           XXX             X
```

Capítulo 4

Cómo estructurar un Programa de BAL

Objetivos Al término de este capítulo usted será capaz de:

- Definir registro en el contexto de la arquitectura System/370,
- Definir una fullword,
- Explique qué efecto, si los hubiere, cada una de las instrucciones siguientes en un registro: BAL, BR, ST, y L.
- Explicar cómo el BAL, BR, ST, y las instrucciones de L se puede utilizar para estructurar un programa en ensamblador,
- Explicar el propósito de PC/370 's macros regs,
- Describir los componentes típicos de la línea principal, la configuración, insumos, procesos, OUTPUT, y Wrapup secciones de un programa estructurado, y de
- Escribir un programa estructurado.

Introducción Algunos dirán que "estructurado BAL" es un oxímoron, que no se puede hacer. BAL tiene una muy conjunto de instrucciones limitadas, es, después de todo, un lenguaje de bajo nivel. La mayoría de los programadores han enseñado a uso de la programación estructurada, una parte importante de los cuales es GO-TO-menos de programación. BAL equivalente a un COBOL o BASIC GOTO es la instrucción de salto, el uso de que es inevitable en la la mayoría de los programas de BAL. Si GO-TO-menos de la programación es parte de sus criterios para la estructura la programación, entonces es cierto que BAL no puede ser "estructurado". Pero el uso selectivo de las normas de la programación estructurada puede recorrer un largo camino hacia la mejora de la legibilidad del programa, mantenimiento y reutilización.

En este capítulo aprenderá cómo estructurar un programa de BAL. En concreto, vamos a aprender a utilizar el siguientes instrucciones: BAL, BR, ST, y L, así como otro uso para EQU.

4.1. CONFIGURACIÓN, PROCESO, y Wrapup Secciones

Vamos a continuar con el programa de TEACH3A.MLC del capítulo anterior. Ese programa produce un lista de instructores titulares de los autos MAESTRO. Nuestro primer paso es dividir el programa en el tres secciones:

- SETUP, o las cosas que suceden una sola vez antes de que los registros son procesados,
- Proceso, o las cosas que suceden una vez por registro, y
- Wrapup, o las cosas que suceden una sola vez después de todos los registros han sido procesados.

La mayoría de programas de negocio siguen una estructura similar.

Se podría utilizar el mismo nombre, o algo similares. Por ejemplo, la instalación es a veces llamado Limpieza (comúnmente abreviado como HSK), De inicialización, o del Banco de Japón (Principio de Empleo), y Wrapup a veces se llama FINALIZACIÓN o EOJ (Fin de Trabajo).

A continuación se muestra la parte de la lógica (sólo) de TEACH3A.MLC. Un cuadro se ha elaborado en torno a las partes de el programa que representan la configuración, PROCESO, y las secciones Wrapup.

97 ⟨*cap 3* 86⟩+≡ ◁86

```
          PRINT NOGEN
*****************************************************************
*         FILENAME: TEACH3A.MLC                                 *
*         AUTHOR : Bill Qualls                                  *
*         SYSTEM : PC/370 R4.2                                  *
*         REMARKS : Produce a list of tenured instructors.      *
*****************************************************************
          START 0
          REGS
BEGIN     BEGIN
          WTO   'TEACH3A ... Begin execution'
          OI    TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                          input from ASCII to EBCDIC
          OI    REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                          output from EBCDIC to ASCII
          OPEN   TEACHERS
          OPEN   REPORT
          PUT    REPORT,HD1
```

```
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
LOOP     EQU    *
         GET    TEACHERS,IREC              Read a single teacher record
         CLI    ITTEN,C'Y'                 Is teacher tenured?
         BNE    LOOP                       No, then skip this record
         MVC    OTID,ITID                  Move teacher ID Nbr to output
         MVC    OTNAME,ITNAME              Move teacher Name to output
         CLC    ITDEG,=CL4'PHD'            Highest degree = PhD?
         BE     YESPHD                     .. Yes, branch
         MVI    OPHD,C'N'                  .. No, Show PhD = 'N'
         B      OTHERS                     .. Branch around YES logic
YESPHD   EQU    *                          Highest degree is PhD, so...
         MVI    OPHD,C'Y'                  Show PhD = 'Y'
OTHERS   EQU    *                          Continue moving other fields...
         MVC    O517,=CL4'517-'            All phone nbrs begin w/ '517-'
         MVC    OTPHONE,ITPHONE            Move phone nbr to output
         MVC    OCRLF,WCRLF                PC/370 ONLY - end line w/ CR/LF
         PUT    REPORT,OREC                Write report line
         B      LOOP
*
*        EOJ processing
*
ATEND    CLOSE TEACHERS
         CLOSE REPORT
         WTO    'TEACH3A ... Teacher list on REPORT.TXT'
         WTO    'TEACH3A ... Normal end of program'
⟨Epilogo 26a⟩
*
*        File definitions
*
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT   DCB    LRECL=62,RECFM=F,MACRF=P,
                DDNAME='REPORT.TXT'
*
*        Miscellaneous field definitions
*
WCRLF    DC     X'0D25'                    PC/370 ONLY - EBCDIC CR/LF
*
*        Input record definition
*
⟨TEACHER IREC, registro de entrada 44⟩
*
*        Output (line) definition
*
```

```
OREC    DS    0CL62
OTID    DS    CL3               Teacher ID nbr
        DC    CL3' '
OTNAME  DS    CL15              Teacher name
        DC    CL4' '
OPHD    DS    CL1               PhD? (Y/N)
        DC    CL5' '
OPHONE  DS    0CL8              Phone nbr
O517    DS    CL4
OTPHONE DS    CL4               Phone nbr
        DC    CL21' '
OCRLF   DS    CL2               PC/370 only - CR/LF
*
*       Headings definitions
*
HD1     DS    0CL62
        DC    CL40'      LIST OF TENURED INSTRUCTORS     '
        DC    CL20' '
        DC    XL2'0D25'
HD2     DS    0CL62
        DC    CL60' '
        DC    XL2'0D25'
HD3     DS    0CL62
        DC    CL40'ID#        Name          PhD?   Phone '
        DC    CL20' '
        DC    XL2'0D25'
HD4     DS    0CL62
        DC    CL40'--   ---------  --   ---'
        DC    CL20' '
        DC    XL2'0D25'
        END   BEGIN
```

La lógica que impulsa a estas secciones de código se refiere comúnmente como la línea principal. Con el fin de para la estructura de nuestro programa, necesitamos una instrucción que nos permitirá invocar subrutinas, que es decir, algo comparable a GOSUB BASIC o PERFORM COBOL, según la cual podemos ir a una subrutina y luego volver una vez que la subrutina ha terminado. En BASIC, cada GOSUB deben tener una retribución; La rentabilidad es una declaración requerida. En COBOL, el regreso está implícito en la final de la designada procedimiento, ya sea un párrafo o sección. La aplicación de BAL de este proceso es más similar a la Básico que en COBOL, en que la devolución debe ser codificado.

El equivalente a un GOSUB BAL es BAL, o Subdivisión y Link. No confunda BAL (Basic Asamblea Language) con BAL (la rama y la instrucción Link)! El equivalente a un retorno BAL es BR, o Subdivisión de Registro. A fin de comprender el BAL y las instrucciones de BR, debemos entender un poco poco acerca de los registros.

Un registro es una zona especial de conservación dentro de la CPU (unidad central de proceso). El tamaño de un registro de dependerá de la de la CPU. Mayoría de los PC, por ejemplo, tienen dos registros de bytes, mientras que el IBM 370 el equipo tiene cuatro registros de bytes. El tamaño del registro de un ordenador se conoce como su tamaño de palabra. Cada registro es capaz de mantener una dirección, y es lógico que cuanto mayor sea el registro, o el tamaño de la palabra, mayor será la dirección que puede contener. (¿Cómo se representa internamente se discutido en detalle más adelante.) El número de registros también varían según el tipo de CPU. El IBM 370 el equipo tiene dieciséis registros. Estos registros están numerados del 0 al 15 y son referidos por número. A pesar de que el PC tiene un número diferente de registros, PC/370 emula un , mainframe, así que el programa como si tuviéramos todos los 16 registros.

Como su nombre indica, la Subdirección de Enlace y la instrucción es una rama, y un poco más. Para ejemplo, la instrucción BAL 10, CONFIGURACIÓN dice a poner la dirección de la instrucción siguiente (el la instrucción inmediatamente después de la instrucción BAL) en el registro 10, y luego a la rama de etiqueta CONFIGURACIÓN.

Entonces, ¿cómo saber cuándo y dónde volver? Bueno, ya registrar 10 tiene la dirección de la la siguiente instrucción, simplemente volver a la dirección a la que registrar 10 está señalando. Para ello, hemos Código BR 10. No confunda esto con una rama de registro 10! La instrucción B 10 se reunirán y se ejecutará, pero con resultados imprevisibles! La instrucción BR dice a la rama a la dirección contenida en el registro establecido. Si usted tiene codificado en C antes, este será un concepto fácil de tomar arriba. Si su única experiencia previa de codificación en BASIC o COBOL, esto llevará algún tiempo para acostumbrarse.

No tiene que utilizar registrar 10. Para repetir, hay dieciséis registros. Usted no puede utilizar registro 13 con PC/370 ya que este es su registro de base. (Es utilizado por PC/370 's BEGIN macro para establecer addressibility para el programa. Esto será discutido en detalle más adelante.) Algunas instrucciones BAL y macros muchos modificar los registros 0 y 1, por lo que es una práctica

común para evitar el uso dichos registros. La mayoría de las instalaciones tendrán un estándar en cuanto a qué registro (s) a utilizar para de ramificación y vínculos.

La nueva versión de nuestro programa, TEACH4A.MLC, del siguiente modo:

101 ⟨ff 40⟩+≡ ◁67 202▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH4A.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : This is a revision of TEACH3A.MLC.           *
*                    Produce list of tenured instructors.       *
*                    How to structure a BAL program.            *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TEACH4A ... Begin execution'
         BAL   10,SETUP
MAIN     EQU   *
         BAL   10,PROCESS
         B     MAIN
ATEND    EQU   *
         BAL   10,WRAPUP
         WTO   'TEACH4A ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU   *
         OI    TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI    REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN TEACHERS
         OPEN REPORT
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         BR    10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
```

```
PROCESS  EQU    *
         GET    TEACHERS,IREC            Read a single teacher record
         CLI    ITTEN,C'Y'               Is teacher tenured?
         BNE    PROCESSX                 No, then skip this record
         MVC    OTID,ITID                Move teacher ID Nbr to output
         MVC    OTNAME,ITNAME            Move teacher Name to output
         CLC    ITDEG,=CL4'PHD'          Highest degree = PhD?
         BE     YESPHD                   .. Yes, branch
         MVI    OPHD,C'N'                .. No, Show PhD = 'N'
         B      OTHERS                   .. Branch around YES logic
YESPHD   EQU      *                      Highest degree is PhD, so...
         MVI      OPHD,C'Y'              Show PhD = 'Y'
OTHERS   EQU      *                      Continue moving other fields...
         MVC      O517,=CL4'517-'        All phone nbrs begin w/ '517-'
         MVC      OTPHONE,ITPHONE        Move phone nbr to output
         MVC      OCRLF,WCRLF            PC/370 ONLY - end line w/ CR/LF
         PUT      REPORT,OREC            Write report line
PROCESSX EQU      *
         BR       10
*****************************************************************
*           WRAPUP - Those things which happen one time only,  *
*                          after all records have been processed.*
*****************************************************************
WRAPUP   EQU      *
         CLOSE TEACHERS
         CLOSE REPORT
         WTO      'TEACH4A ... Teacher list on REPORT.TXT'
         BR       10
(Remainder of program is the same as TEACH3A.MLC.)
```

4.1.1. La Revisión de rutina PROCESO ...

Programa de TEACH4A.MLC contiene dos violaciónes más grave de las normas de estructura de programación. En primer lugar, cada módulo o función debe tener un único objetivo bien definido. Esta es la no es el caso con nuestro módulo de proceso: este módulo se lee un registro, se evalúa que el historial de acuerdo a los criterios de extracción, formatos de una línea de informe, y escribe una línea de informe:

```
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS EQU    *
                                                        READ
         GET   TEACHERS,IREC      Read a single teacher record
                                                        EXTRACT
         CLI   ITTEN,C'Y'         Is teacher tenured?
         BNE   PROCESSX           No, then skip this record
                                                        FORMAT
         MVC   OTID,ITID          Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME      Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE    YESPHD             .. Yes, branch
         MVI   OPHD,C'N'          .. No, Show PhD = 'N'
         B     OTHERS             .. Branch around YES logic
YESPHD   EQU   *                  Highest degree is PhD, so...
         MVI   OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU   *                  Continue moving other fields...
         MVC   O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE    Move phone nbr to output
         MVC   OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
                                                        WRITE
         PUT   REPORT,OREC        Write report line
PROCESSX EQU   *
         BR    10

************************
```

Un módulo con un solo propósito, bien definido se dice que ser coherente. El programado

La nueva sección de proceso se convierte en:

```
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
```

```
*****************************************************************
PROCESS  EQU    *
         BAL    10,READ            Read a single teacher record
         CLI    ITTEN,C'Y'         Is teacher tenured?
         BNE    PROCESSX           No, then skip this record
         BAL    10,FORMAT          Otherwise, format report line
         BAL    10,WRITE           and write report line
PROCESSX EQU    *
         BR     10
*****************************************************************
*        READ A RECORD                                          *
*****************************************************************
READ     EQU    *
         GET    TEACHERS,IREC      Read a single teacher record
         BR     10
*****************************************************************
*        FORMAT A LINE                                          *
*****************************************************************
FORMAT   EQU    *
         MVC    OTID,ITID          Move teacher ID Nbr to output
         MVC    OTNAME,ITNAME      Move teacher Name to output
         CLC    ITDEG,=CL4'PHD'    Highest degree = PhD?
         BE     YESPHD             .. Yes, branch
         MVI    OPHD,C'N'          .. No, Show PhD = 'N'
         B      OTHERS             .. Branch around YES logic
YESPHD   EQU    *                  Highest degree is PhD, so...
         MVI    OPHD,C'Y'          Show PhD = 'Y'
OTHERS   EQU    *                  Continue moving other fields...
         MVC    O517,=CL4'517-'    All phone nbrs begin w/ '517-'
         MVC    OTPHONE,ITPHONE    Move phone nbr to output
         MVC    OCRLF,WCRLF        PC/370 ONLY - end line w/ CR/LF
*****************************************************************
*        WRITE A LINE                                           *
*****************************************************************
WRITE    EQU    *
         PUT    REPORT,OREC        Write report line
         BR     10
```

Pero esto no funcionará! Cuando se invoca un módulo, la mayoría de idiomas (como el COBOL y PERFORM BASIC GOSUB) utilizan una "pila" a fin de que las invocaciones anidadas como ésta. Esta no es la caso de BAL. El ejemplo anterior causará un bucle infinito. ¿Puedes ver por qué? Echemos un más de cerca.

```
(1)          BAL 10,PROCESS
(2)          B   MAIN              :
```

```
(3) PROCESS EQU  *
(4)          BAL 10,READ
             CLI ITTEN,C'Y'
             BNE PROCESSX
(5)          BAL 10,FORMAT
(6)          BAL 10,WRITE
    PROCESSX EQU *
             BR
```

En el momento en que la instrucción (1) se ejecuta, registro de 10 puntos a la siguiente instrucción, la instrucción (2). La intención es que cuando llegamos al final del módulo de proceso, la instrucción (7) hará que flujo de control para la transferencia a la instrucción inmediatamente después de su invocación, que se instrucción (2). Pero esto no va a suceder ....

Sigamos. Como resultado de la instrucción (1), el flujo de control va a la instrucción (3). Se trata de un única etiqueta, de modo que el flujo de control de las caídas a través de la instrucción (4). Cuando la instrucción (4) se ejecuta, registro de 10 puntos a la siguiente instrucción, la enseñanza (5). Estamos en problemas ya: el valor de registro de 10 ha cambiado! ¿Cómo vamos a volver a la instrucción (2)? La BR 10 en la final de la LEA rutina nos llevará de vuelta a la enseñanza (5). Lo mismo sucede con las instrucciones (5) y (6). Cuando se ejecuta la instrucción (6), registro de 10 puntos a la instrucción (7). La BR 10 en la final de la rutina de escribir va a traernos de vuelta a la instrucción (7). Instrucción (7) dice que para ir a donde registrar 10 está señalando. Pero registrar 10 apunta a la instrucción (7), que dice que para ir a la donde registrar 10 está señalando .... Bucle sin fin.

Una solución a este problema es utilizar un registro diferente para cada rama-y-Link. Pero este la solución no es satisfactoria por razones obvias: usted se quedará sin registros. (Hay sólo dieciséis años.) Y tienes que ser muy cuidadosos para invocar una rutina (con BAL) utilizando el mismo registro de que que la rutina volverá (con BR).

La solución a este problema es:

Guardar la dirección de retorno para cada rutina inmediatamente después de la entrada, y Restaurar la dirección para el registro adecuado inmediatamente antes de salir.

De esta manera podemos utilizar el mismo registro para todas las Sucursales y enlaces.

Para guardar la dirección de retorno, que está en un registro, utilizamos el ST (la tienda) la instrucción. ¿Dónde vamos a salvarla? En un campo que vamos a definir y dejar de lado sólo para este propósito. Recordemos el IBM 370 el equipo tiene cuatro registros de bytes, y el tamaño del registro de un ordenador se conoce como su palabra tamaño. Así que vamos a utilizar un controlador de dominio con el tipo de campo de fullword para guardar el contenido del registro. Para ejemplo:

```
LEA EQU *
```

```
                    ST 10, SVREAD
dónde
          SVREAD DC F'0 '
```

(Tenga en cuenta que la instrucción de la tienda es uno de los pocos elegidos en el primer operando es el campo de envío, la no el campo de recepción. Es al revés de la mayoría de las instrucciones.)

Para restaurar el valor al registro, utilizamos la L (de carga) la instrucción. Por ejemplo:

```
L 10, SVREAD
                    BR 10
```

Esta es una técnica simple pero efectivo para la asignación de módulos y se utiliza en este libro.

Nota: Esta técnica no apoyará invocación recursiva, pero la necesidad de tales transformación es poco común en las aplicaciones empresariales. Si usted quiere hacer llamadas recursivas, se le necesidad de definir una pila, y el código del equivalente de push y pop operaciones de pila. No hemos BAL aprendido lo suficiente como para hacer eso todavía.

La siguiente es una lista parcial de TEACH4B.MLC, la nueva versión del programa.

106 ⟨*search..., write REPORT from OREC, no line counting, SVWRITE return* 106⟩≡ (107 113 157 207 211 357 42

```
WRITE   EQU     *
        ST      10,SVWRITE
        PUT     REPORT,OREC             Write report line
        L       10,SVWRITE
        BR      10
```

107 ⟨ *TEACH4B.MLC* 107 ⟩≡

```
          PRINT NOGEN
*****************************************************************
*         FILENAME: TEACH4B.MLC                                 *
*         AUTHOR : Bill Qualls                                  *
*         SYSTEM : PC/370 R4.2                                  *
*         REMARKS : This is a revision of TEACH4A.MLC.          *
*                     Produce list of tenured instructors.      *
*                     How to structure a BAL program.           *
*****************************************************************
          START 0
          REGS
BEGIN     BEGIN
          WTO    'TEACH4B ... Begin execution'
          BAL    10,SETUP
MAIN      EQU    *
          BAL    10,PROCESS
          B      MAIN
ATEND     EQU    *
          BAL    10,WRAPUP
          WTO    'TEACH4B ... Normal end of program'
          RETURN
*****************************************************************
*         SETUP - Those things which happen one time only,      *
*                 before any records are processed.             *
*****************************************************************
SETUP     EQU   *
          ST    10,SVSETUP
          OI    TEACHERS+10,X'08'         PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
          OI    REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
          OPEN TEACHERS
          OPEN REPORT
          BAL   10,HDGS
          L     10,SVSETUP
          BR    10
*****************************************************************
*         HDGS - Print headings.                                *
*****************************************************************
HDGS      EQU   *
          ST    10,SVHDGS
          PUT   REPORT,HD1
          PUT   REPORT,HD2
          PUT   REPORT,HD3
          PUT   REPORT,HD4
```

```
         L     10,SVHDGS
         BR    10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU    *
         ST    10,SVPROC
         BAL   10,READ
         CLI   ITTEN,C'Y'               Is teacher tenured?
         BNE   PROCESSX                 No, then skip this record
         BAL   10,FORMAT                Otherwise format a line
         BAL   10,WRITE                 ...and write it
PROCESSX EQU   *
         L     10,SVPROC
         BR    10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
READ     EQU   *
         ST    10,SVREAD
         GET   TEACHERS,IREC            Read a single teacher record
         L     10,SVREAD
         BR    10
*****************************************************************
*        FORMAT - Format a single detail line.                  *
*****************************************************************
FORMAT   EQU   *
         ST    10,SVFORM
         MVC   OTID,ITID                Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME            Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'          Highest degree = PhD?
         BE    YESPHD                   .. Yes, branch
         MVI   OPHD,C'N'                .. No, Show PhD = 'N'
         B     OTHERS                   .. Branch around YES logic
YESPHD   EQU     *                      Highest degree is PhD, so...
         MVI     OPHD,C'Y'              Show PhD = 'Y'
OTHERS   EQU     *                      Continue moving other fields...
         MVC     O517,=CL4'517-'        All phone nbrs begin w/ '517-'
         MVC     OTPHONE,ITPHONE        Move phone nbr to output
         MVC     OCRLF,WCRLF            PC/370 ONLY - end line w/ CR/LF
         L       10,SVFORM
         BR      10
*****************************************************************
*           WRITE - Write a single detail line.                 *
*****************************************************************
```

⟨*search..., write REPORT from OREC, no line counting, SVWRITE return* 106⟩

```
*****************************************************************
*           WRAPUP - Those things which happen one time only,  *
*                        after all records have been processed.*
*****************************************************************
WRAPUP   EQU     *
         ST      10,SVWRAP
         CLOSE TEACHERS
         CLOSE REPORT
         WTO     'TEACH4B ... Teacher list on REPORT.TXT'
         L       10,SVWRAP
         BR      10
*****************************************************************
*           Literals, if any, will go here                      *
*****************************************************************
         LTORG
*****************************************************************
*           File definitions                                    *
*****************************************************************
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT   DCB     LRECL=62,RECFM=F,MACRF=P,
                 DDNAME='REPORT.TXT'
*****************************************************************
*           RETURN ADDRESSES                                    *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
       (Remainder of program is the same as TEACH3A.MLC)
```

109 ⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩≡ (107 113 157 164

```
SVSETUP  DC      F'0'                    SETUP
SVHDGS   DC      F'0'                    HDGS
SVPROC   DC      F'0'                    PROCESS
SVREAD   DC      F'0'                    READ
SVFORM   DC      F'0'                    FORMAT
SVWRITE  DC      F'0'                    WRITE
SVWRAP   DC      F'0'                    WRAPUP
```

4.1.2. La Revisión de rutina Leer más ...

Anteriormente se dijo que había dos violaciónes de las normas de la programación estructurada. La segunda regla es cada módulo debe tener una sola entrada y una salida única. Esto no es del todo posible, dadas nuestro conjunto de instrucciones limitadas, pero algunas técnicas son mejores que otros. Recordemos que la EODAD parámetro de la macro DCB indica que el programa debe ir cuando la entrada correspondiente archivo llega al final de su archivo. En la TEACH4B.MLC EODAD envía el control de nuevo a la línea principal (a

atEnd); es decir, que sale de la rutina de LEA de otros que la salida normal (la BR al final de la LEA), y la regresa a la línea principal sin tener que volver a la rutina de proceso que se invoca en el LEA primer lugar. Esta es una violación de la ventanilla única, la regla de salida única.

Para el beneficio de los programadores de COBOL, COBOL consideran el equivalente a lo que acabamos de visita:

```
PROCEDURE DIVISION.
MAINLINE.
    PERFORM SETUP.
MAIN-LOOP.
    PERFORM PROCESS-A-RECORD.
    GO TO MAIN-LOOP.
AT-END-OF-FILE.
    PERFORM WRAPUP.
    GOBACK.
PROCESS-A-RECORD.
    PERFORM READ-A-RECORD.
    IF extract-criteria-met
       PERFORM FORMAT-A-RECORD
       PERFORM WRITE-A-RECORD.
READ-A-RECORD.
    READ TEACHER-FILE INTO WS-TEACHER-RECORD
        AT END GO TO AT-END-OF-FILE.
```

Es más común en un programa en COBOL para utilizar un fin de archivo interruptor. También es común que incluya, un resanador leer como la última instrucción dentro de la configuración, y otra leer como la última instrucción de el párrafo PROCESO. Esto se ilustra en el segmento de programa siguiente.

```
WORKING-STORAGE SECTION.
01 WS-MISC.
    05 END-OF-FILE-SW        PIC X(1) VALUE 'N'.
         88 END-OF-FILE               VALUE 'Y'.
    :
    :
PROCEDURE DIVISION.
MAINLINE.
    PERFORM SETUP.
    PERFORM PROCESS-A-RECORD
       UNTIL END-OF-FILE.
    PERFORM WRAPUP.
    GOBACK.
SETUP.
    :
    :
    PERFORM READ-A-RECORD.
```

```
PROCESS-A-RECORD.
    IF extract-criteria-met
       PERFORM FORMAT-A-RECORD
       PERFORM WRITE-A-RECORD.
    PERFORM READ-A-RECORD.
READ-A-RECORD.
    READ TEACHER-FILE INTO WS-TEACHER-RECORD
        AT END MOVE 'Y' TO END-OF-FILE-SW.
FORMAT-A-RECORD.
    :
```

No podemos igualar este en BAL pero podemos acercarnos. El BAL problema es que no tiene

111 ⟨*search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP* 111⟩≡ (113 157 164 187 207 211 223 232 241

```
         BAL    R10,SETUP
MAIN     EQU    *
         CLI    EOFSW,C'Y'
         BE     EOJ
         BAL    R10,PROCESS
         B      MAIN
EOJ      EQU    *
         BAL    R10,WRAPUP

           :
           :
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
SETUP    EQU   *
         :
         :
         BAL   10,READ
         L     10,SVSETUP
         BR    10
         :
         :
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         CLI   ITTEN,C'Y'          Is teacher tenured?
         BNE   PROC2               No, then skip this record
         BAL   R10,FORMAT          Otherwise format a line
         BAL   R10,WRITE           ...and write it
PROC2    EQU   *
```

```
          BAL   R10,READ           Read next
PROCESSX  EQU   *
          L     10,SVPROC
          BR    10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
```

112 ⟨*search..., read TEACHERS into IREC, EOFSW, SVREAD return* 112⟩≡ (113 187)

```
READ      EQU   *
          ST    R10,SVREAD
          GET   TEACHERS,IREC              Read a single teacher record
          B     READX
ATEND     EQU   *
          MVI   EOFSW,C'Y'
READX     EQU   *
          L     R10,SVREAD
          BR    R10

              :
              :
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFSW     DC    CL1'N'             End of file? (Y/N)
```

Un comentario más antes de presentar nuestra versión final del programa. Si nos fijamos en la cruz lista de referencias producidas por PC/370 (o cualquier otro ensamblador de 370) no verá ninguna referencia para registrar 10 (o cualquier otro registro). (La lista de referencia recíproca con PC/370 se pgmname.PRN, que es creado por A370.) Uso de los registros es tan común dentro de los programas que es muy conveniente disponer de ellas incluidas en la lista de referencia cruzada para el escritorio de comprobación y la depuración de propósitos. Por otra parte, queremos distinguir entre 10 (el registro) de 10 (el número o la dirección, o la longitud, o el desplazamiento). Esto se hace con el EQU (Equate) la instrucción. Por ejemplo: R10 EQU 10

Esto es tan común, que (PC/370 y la mayoría de las instalaciones) tienen una macro que equipara todos los registros para usted. He visto esta macro llamada EQUATE, REGEQU, y otros. PC/370 llama a esta REGs macro. Ello está codificado inmediatamente después del comando START. Cuando el preprocesador de macros, M370, encuentros esta macro, se genera una instrucción EQU similar para todos los 16 registros. (Usted puede ver esta viendo el pgmname.ALC archivo creado por M370.) Vamos a utilizar en TEACH4C y todos los programas posteriores. (Hasta a este punto, hemos incluido la macro REGs en nuestros programas, pero era opcional desde que hicimos no hacer referencia a cualquier registros.) Podemos utilizar R10 en lugar de 10 cuando nos referimos

a registrar 10. R10 aparecerá en la lista de referencia cruzada producida por el ensamblador.

Nuestra versión final del programa, TEACH4C.MLC, sigue.

113 ⟨*TEACH4C.MLC* 113⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH4C.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : This is a revision of TEACH4B.MLC.           *
*                    Produce list of tenured instructors.       *
*                    How to structure a BAL program.            *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'TEACH4C ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'TEACH4C ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     TEACHERS+10,X'08'          PC/370 ONLY - Convert all
*                                          input from ASCII to EBCDIC
         OI     REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                          output from EBCDIC to ASCII
         OPEN TEACHERS
         OPEN REPORT
         BAL    R10,HDGS
         BAL    R10,READ                   Priming read
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
```

```
         L      R10,SVHDGS
         BR     R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS  EQU    *
         ST     R10,SVPROC
         CLI    ITTEN,C'Y'                Is teacher tenured?
         BNE    PROC2                     No, then skip this record
         BAL    R10,FORMAT                Otherwise format a line
         BAL    R10,WRITE                 ...and write it
PROC2    EQU    *
         BAL    R10,READ                  Read next
PROCESSX EQU    *
         L      R10,SVPROC
         BR     R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
⟨search..., read TEACHERS into IREC, EOFSW, SVREAD return 112⟩
*****************************************************************
*        FORMAT - Format a single detail line.                  *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORM
         MVC   OTID,ITID                 Move teacher ID Nbr to output
         MVC   OTNAME,ITNAME             Move teacher Name to output
         CLC   ITDEG,=CL4'PHD'           Highest degree = PhD?
         BE    YESPHD                    .. Yes, branch
         MVI   OPHD,C'N'                 .. No, Show PhD = 'N'
         B     OTHERS                    .. Branch around YES logic
YESPHD   EQU   *                         Highest degree is PhD, so...
         MVI   OPHD,C'Y'                 Show PhD = 'Y'
OTHERS   EQU   *                         Continue moving other fields...
         MVC   O517,=CL4'517-'           All phone nbrs begin w/ '517-'
         MVC   OTPHONE,ITPHONE           Move phone nbr to output
         MVC   OCRLF,WCRLF               PC/370 ONLY - end line w/ CR/LF
         L     R10,SVFORM
         BR    R10
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
⟨search..., write REPORT from OREC, no line counting, SVWRITE return 106⟩
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
```

```
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH4C ... Teacher list on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
⟨TEACHERS DCB, EODAD=ATEND 26c⟩
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                   End of file? (Y/N)
*****************************************************************
*        Input record definition                                *
*****************************************************************
⟨TEACHER IREC, registro de entrada 44⟩
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62
OTID     DS    CL3                      Teacher ID nbr
         DC    CL3' '
OTNAME   DS    CL15                     Teacher name
         DC    CL4' '
OPHD     DS    CL1                      PhD? (Y/N)
         DC    CL5' '
OPHONE   DS    0CL8                     Phone nbr
O517     DS    CL4
OTPHONE  DS    CL4                      Phone nbr
         DC    CL21' '
OCRLF    DS    CL2                      PC/370 only - CR/LF
```

```
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
HD1      DS    0CL62
         DC    CL40'         LIST OF TENURED INSTRUCTORS                    '
         DC    CL20' '
         DC    XL2'0D25'
HD2      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL40'ID#              Name                    PhD?        Phone '
         DC    CL20' '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'-- ----------------  --     ----  '
         DC    CL20' '
         DC    XL2'0D25'
         END   BEGIN
```

Si se compara este programa (TEACH4C.MLC) del capítulo 3 (versión TEACH3A.MLC), la inicial reacción puede ser que hemos logrado hacer una montaña de un grano de arena. A primera vista, el nuevo programa parece mucho más complicado. Pero pronto se hará evidente que programas estructurados de esta manera son mucho más fáciles de mantener. Además, el uso de las pequeñas, coherente módulos aumenta la reusabilidad del código: es mucho más fácil de usar partes de un programa en la otro sin modificación. Vamos a utilizar esta estructura en el resto del libro. Usted puede estar sorprendido de lo poco que tenemos que cambiar a medida que se introducen nuevos conceptos!

4.2. Ejercicios

1. 1. Verdadero o falso.

   a) El tamaño de los registros de un equipo se conoce como el tamaño de la palabra. F T

   b) El equipo System/370 dieciséis registros se numeran del 1 al 16. F T

   c) PC/370utiliza registro de 13, como su registro de base. F T

   d) La instrucción de la tienda (ST) funciona como instrucciones BAL mayoría de los otros, es decir, la F T el primer operando es el campo de recepción.

   e) La instrucción de carga (L) funciona como instrucciones BAL mayoría de los otros, es decir, la F T el primer operando es el campo de recepción.

f) La instrucción de carga (L) carga un registro en un fullword. F T

g) Usamos el registro de la sucursal (BR) la orden de regresar de una subrutina. F T

h) Un registro diferente debe ser utilizado para invocar a cada subrutina. F T

i) Las técnicas mostradas en este capítulo no son compatibles con la recursividad. F T

j) Para maximizar la cohesión de medios para poner tanto código en una subrutina único como es F T posible.

k) Cada módulo debe tener una sola entrada y una salida única. F T

l) SETUP las cosas que suceden una sola vez antes de que los registros se F T procesados.

m) La rutina de LEA se invoca desde el SETUP y rutinas PROCESO. F T

2. Haga el ejercicio 14, 15 o 16 del capítulo 3. Su programa debe estar estructurado, haciendo pleno uso de los conceptos mostrados en este capítulo.

3. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Escribir un programa de que la lista de esos temas en la herramienta de archivo en el que la cantidad en la mano es igual o inferior al cantidad mínima. No envolturas de lista, que se indican por un precio de venta de cero. El informe debería aparecer como sigue:

```
1         2          3         4        5         6 123456789012345678901234567890
                     SMALL TOWN HARDWARE STORE
ITEMS TO BE ORDERED
TID        Description           Cost      Sell    QOH    MIN
---   ---------------------    ------    ------ -----   -----
XXX   XXXXXXXXXXXXXXXXXXXX    XXX.XX    XXX.XX   XXX    XXX
XXX   XXXXXXXXXXXXXXXXXXXX    XXX.XX    XXX.XX   XXX    XXX
XXX   XXXXXXXXXXXXXXXXXXXX    XXX.XX    XXX.XX   XXX    XXX
```

Costo y venta de precio son almacenados sin decimales, pero se debe imprimir para que ellos hacen mostrar un decimal. Por ejemplo, un valor almacenado de 00425 se debe imprimir en 004,25. No estar preocupado por ceros a la izquierda en este momento. Su programa debe estar estructurado, haciendo pleno uso de los conceptos mostrados en este capítulo.

4. Ejercicios 4.

Capítulo 5

Representación de datos

Objetivos Al término de este capítulo usted será capaz de:

- Describir los sistemas de numeración binario y hexadecimal,
- Dado un número en una de base 2, 10, o 16, convertir ese número a su equivalente en la otra bases,
- Además de realizar con un número binario y hexadecimal, y
- Demostrar cómo el personaje se almacena internamente usando el EBCDIC o ASCII cotejo secuencias.

Introducción Lo hemos hecho sobre el ensamblador lo que podamos sin aprender más acerca de cómo los datos se representados internamente. En este capítulo hablaremos de los sistemas de número binario y hexadecimal. Pero primero, tenemos que tomar una breve mirada a nuestro sistema de numeración decimal ....

5.1. Una mirada más cercana a nuestro Sistema de numeración decimal

Hora de que algún tipo de revisión aritmética muy básica. Esto puede tomar un tiempo de vuelta! Tenga en cuenta la número 1956. Sabemos que este es un número decimal válido, pero ¿por qué? Es un número decimal válido porque está compuesto de dígitos es válida para los decimales, o en base 10, sistema numérico, en concreto, 0 al 9 inclusive. Es decir, el sistema de base 10 el número consta de diez dígitos. Recordemos que el 10 no es un dígito, pero es un número decimal válido formado por los dígitos 1 y 0.

Consideremos de nuevo el número 1956. Sabemos que esta cifra sea un mil novecientos cincuenta y seis, pero ¿por qué? Veamos esto más de cerca. Recordemos:

```
                                    1956
                                     ::::
                                     :::::.. 6 x 1 = 6
                                     :::.... 5 x 10 = 50
                                     ::..... 9 x 100 = 900
                                     :...... 1 x 1000 = 1000
                                                          1956
```

Además, recuerda:

```
10 0    1 =
10 1   10 =
10 2  100 =
10 3 1000 =
```

Y así podemos volver a escribir lo anterior como:

```
1956                              ::::                                  10 0
```

Así vemos que para un número en base 10, cada dígito se multiplica por un poder sucesivos de diez.

¿Cuál es el número más alto con cuatro dígitos decimales? Respuesta: 9999. Sabemos esto porque 9 es la cifra más alta permitida en el sistema de numeración en base 10, y para obtener el mayor número de decimales con cuatro dígitos, simplemente use este dígitos en todas las cuatro posiciones.

Del mismo modo, el menor número decimal con cuatro dígitos es 0000, ya que 0 es la más baja de dos dígitos permitido en el sistema de numeración en base 10, y para obtener el menor número con cuatro dígitos decimales, simplemente uso de este dígito en todas las cuatro posiciones.

Así pues, en resumen,

- El sistema de base 10 consta de diez dígitos. Ellos son del 0 al 9 inclusive.
- El valor de un número se obtiene multiplicando cada dígito por una potencia sucesivos de diez años y sumando estos productos.
- Para obtener el mayor número en un número determinado de dígitos, utiliza el dígito más alto (9) en todas las posiciones. Del mismo modo, para obtener el número más bajo en un número determinado de dígitos, utiliza el menor de dos dígitos (0) en todos los posiciones.

Estos mismos principios se utilizan como hablamos de otros sistemas de números!

5.2. El sistema binario Número

El sistema numérico binario es también conocido como el sistema numérico de base 2. Así como el sistema de base 10 tiene 10 dígitos, la base 2 sistema

tiene 2 dígitos. Ellos son 0 y 1. Así como 10 no era válido en un dígito el sistema de base 10, 2 no es un dígito válido en la base 2 del sistema.

Así vemos que un número binario es un compuesto de ceros y unos solamente. 10110 es un binario válido número, pero 10.120 no es válido. Tenga en cuenta que 10110 es un número decimal válido. Cuando la base está no es evidente, a menudo utilizamos un subíndice para indicar la base. Por ejemplo, 10110 2 frente a 10.110 10.

El sistema numérico binario es el sistema utilizado por todos los ordenadores para representar datos. La razón de ello es que un dígito binario es fácilmente representada por vía electrónica, como por la presencia o ausencia de corriente.

Cada dígito binario se denomina un poco. Bit es una abreviatura de dígito binario.

5.2.1. De binario a decimal de conversión

Utilizando la misma técnica que se utiliza con el sistema de numeración en base 10, el valor de un binario número se puede encontrar multiplicando cada dígito por una potencia sucesiva de dos y sumando estos productos. Por ejemplo,

```
                        10110
                         :::::
20                          ::::::: 0 x = 0 x 1 = 0
21                          ::::::: 1 x = 1 x 2 = 2
22                          ::::::: 1 x = 1 x 4 = 4
23                          ::::::: 0 x = 0 x 8 = 0
24                          ::::::: 1 x = 1 x 16 = 16
                                                  22
```

Así vemos que 10.110 2 = 22 10

Usted probarlo ...

1. 1101 2 = _____ 10
2. 10010 2 = _____ 10
3. 110111 2 = _____ 10

5.3. El EBCDIC y ASCII para colación de secuencias

de Un byte es un conjunto de bits. Generalmente hay 8 bits en un byte, una excepción de algunos de los primeros dispositivos de teletipo que había 6 o 7 bits en un byte. Para nuestros fines, siempre vamos a asumir de 8 bits por byte.

Una pregunta natural es, pues, "¿Cuál es el valor decimal máximo representable por un solo byte?" Recuerde que en la discusión del sistema de numeración en base 10, para obtener el mayor número en un fijo número de dígitos,

utiliza la más alta de dos dígitos en todas las posiciones. El mayor de dos dígitos en la base del número 2 sistema es 1, por lo que la mayor base de 2 número de 8 dígitos es 11111111.

```
                        11111111
                         ::::::::
20                          ::::::::.. 1 x = 1 x 1 = 1
21                          :::::::... 1 x = 1 x 2 = 2
22                          ::::::.... 1 x = 1 x 4 = 4
23                          :::::..... 1 x = 1 x 8 = 8
24                          ::::...... 1 x = 1 x 16 = 16
25                          :::....... 1 x = 1 x 32 = 32
26                          ::........ 1 x = 1 x 64 = 64
                                                      255
```

Así vemos que el máximo número decimal representable por 8 bits o 1 byte es 255. Del mismo modo, el menor número de decimales que puede representarse por 8 bits o 1 byte es 0 (00000000 2).

Hay, por tanto, un total de 256 valores posibles (de 0 a 255) que puede representarse por un byte.

Este hecho es importante, pero aún no responde la pregunta, "¿Cómo funciona un ordenador, que los procesos de bits (que están encendidas o apagadas solamente) representan los datos? "La respuesta a la cuestión radica en la utilización de códigos.

Es posible que haya jugado con códigos secretos, cuando eras más joven. Por ejemplo, usted puede tener utiliza el siguiente:

```
ABCDEFGHIYZ
1 2 3 4 5 6 7 8 9 25 26
```

El uso de este código, si tengo que enviar la col palabra, me gustaría enviar 3-1-2-2-1-7-5 lugar, y nadie sabía lo que estaba diciendo. (Esto es comúnmente referido como un código, pero en realidad es lo que se conoce como cifrado de sustitución. Un código es realmente algo diferente.)

Con el fin de representar los datos, un sistema similar de codificación se utilizan para asignar un valor decimal (de 0 a 255 inclusive) a cada carácter. Hay dos regímenes de tal codificación. La primera, EBCDIC, o Extended Binary Coded Decimal Interchange Code, es el código utilizado por los mainframes de IBM (y de por lo tanto PC/370). Una porción de código que se muestra aquí:

```
A. .193 G. .199 M. .212 S. .226 Y. .232 4 .. 244          B. .194 H. .200 N. .213 T. .22
```

ASCII, o Código Estándar Americano para el Intercambio de Información, es el código utilizado por los no-IBM los mainframes, y todos los ordenadores. Una porción de código que se muestra aquí:

```
A. .65 G. .71 M. .77 S. .83 Y. .89 4 .. 52        B. .66 H. .72 N. .78 T. .84 Z. .90 5
```

Estos códigos de determinar la secuencia de clasificación, o la secuencia de clasificación, de los personajes. Tenga en cuenta que cuando se utiliza EBCDIC, cartas vienen antes de los números, mientras que en ASCII, números vienen antes de las letras. Del mismo modo, cuando se utiliza EBCDIC, minúsculas antes de venir letras mayúsculas, mientras que en ASCII, letras mayúsculas vienen antes de minúsculas.

5.3.1. Decimal a binario de conversión

Para convertir un número decimal a su equivalente binario, restar la mayor potencia de dos que se inferior o igual al número. Continúe hasta que la diferencia es igual a cero. Los poderes de dos que se restaron será de 1 en el número binario. Todos los dígitos de otros será 0. Para ejemplo:

```
55                              32 = 2 5 -> el quinto dígito es 1 *              -
```

Otro ejemplo:

```
200                             128 = 2 7 -> el séptimo dígito es 1              -1
```

**Tenga en cuenta que las referencias a la colocación de dos dígitos son relativos a cero, que es quinto dígito es en realidad el sexto dígito de la derecha.*

Usted probarlo ...

```
4. 2010 = __________ 2
5. 8710 = __________ 2
6. 14610 = __________ 2
```

El sistema numérico hexadecimal Trabajar con números binarios puede ser muy tedioso: los números son largas y son propensos a errores. El hexadecimal o base 16, sistema de numeración se utiliza comúnmente en lugar de números binarios.

Así como el sistema de base 10 tiene 10 dígitos, y el sistema de base 2 tiene 2 dígitos, el sistema de base 16 16 dígitos. Se extienden en valor de 0 a 15 inclusive. Pero 15 no puede ser un dígito, ya que sería confundirse con un 1 y un 5. Así que para evitar tal confusión, usamos las letras AF para los dígitos 10-15. Los dígitos hexadecimales completa son los siguientes:

```
Hex dígito decimal binario
0 0 0000
```

```
1 1 0001
2 2 0010
3 3 0011
4 4 0100
5 5 0101
6 6 0110
7 7 0111
8 8 1000
9 9 1001
Un 10 1010
B 11 1011
C 12 1100
D 13 1101
E 14 1110
F 15 1111
```

Para ser eficaz en ensamblador, debe confirmar este cuadro a la memoria.

Tenga en cuenta que cada dígito hexadecimal utiliza cuatro bits. Por lo tanto, dos dígitos hexadecimales son necesarias para indicar el valor de un solo byte (ocho bits).

Como antes, existe la posibilidad de confusión sobre la base de un número. El número puede 2A7F sólo un número hexadecimal, pero 10110 es un binario válido, decimales, y el número hexadecimal!

Cuando la base no es evidente, se utiliza un subíndice de 16 para indicar hexadecimal, por ejemplo,

```
10.110 16.
```

5.3.2. Hexadecimal a decimal de conversión

Utilizando la misma técnica que se utilizó con la base 10 y base 2 sistemas de numeración, el valor de de un número hexadecimal se puede encontrar multiplicando cada dígito por una potencia de dieciséis años sucesivos y el resumen de estos productos. Por ejemplo,

```
                        2A7F
                         ::::
16 0                          :::::::. F x = 15 x 1 = 15
16 1                          :::::::. 7 x = 7 x 16 = 112
16 2                          :::::::. A x = 10 x 256 = 2560
16 3                          :::::::. 2 x = 2 x 4096 = 8192
```

Así vemos que 2A7F16 = 10879 10

Usted probarlo ...

```
7. 6D16 = ______ 10
8. A516 = ______ 10
9. 2FE16 = ______ 10
```

5.3.3. Decimales a hexadecimal de conversión

El proceso de convertir un número decimal a su equivalente en hexadecimal es similar al proceso de de convertir un número decimal a su equivalente binario: restar la más alta múltiples de un poder de 16, que es inferior o igual al número. Continúe hasta que la diferencia es igual a cero. Los múltiplos de potencias de 16, que se restará a ser los dígitos en el hexadecimal número. Todos los dígitos de otros será 0. Por ejemplo, para encontrar el equivalente en hexadecimal a 748:

```
748                                    2 x 16 2 -> el segundo dígito es 2
```

En la tabla siguiente se simplificará el proceso de conversión:

```
tercer dígito segundo dígito cero dígitos primer dígito        hexadecimal decimal deci
```

Usted probarlo ...

1. 10. 31210 = _____ 16
2. 11. 70810 = _____ 16
3. 12. 77810 = _____ 16

5.3.4. Hexadecimal a binario de conversión

Tendrá que ser capaz de convertir un número hexadecimal a su equivalente binario y viceversa. Esto es muy fácil de hacer.

Para convertir un número hexadecimal a su equivalente binario, simplemente reemplaza cada dígito hexadecimal con su equivalente en cuatro bits según la tabla de arriba. Por ejemplo, para convertir 5E16 a binario, sustituir 5 en 0101 y E 1110 con dar 01.011.110 2.

Usted probarlo ...

```
13. 4A16 = ____________ 2
14. B1716 = ____________ 2
```

5.3.5. De binario a hexadecimal de conversión

Para convertir un número binario a su equivalente en hexadecimal, simplemente divida los bits en grupos de cuatro. Inicio de la derecha, y agregar ceros a la izquierda extra si es necesario por lo que el número total de bits es un múltiplo de cuatro. A continuación, sustituya cada grupo de cuatro con un solo dígito hexadecimal. Para ejemplo, para convertir 10010010110111 2 a hexadecimal, volver a escribir como | 10 | 0100 | 1011 | 0111 |, añadir dos ceros a la

izquierda dando | 0010 | 0100 | 1011 | 0111 |, entonces reemplazar cada cuatro cuerdas con un poco de su equivalente en hexadecimal de dos dígitos, dando 24B716.

Usted probarlo ... 15. 10111001001 2 = _____ 16

16. 10011010100100 2 = _____ 16

Muestra del Programa Hemos visto que cuando la secuencia de clasificación EBCDIC se utiliza la letra A es equivalente a un 193 decimal. Este valor (193) también es igual a 11.000.001 2 o C116.

Esto se muestra en el siguiente programa:

```
A:\MIN>type data5a.mlc
         PRINT NOGEN
         START 0
BEGIN    BEGIN
         MVI   LETTER,W
         WTO   MESSAGE
         MVI   LETTER,X
         WTO   MESSAGE
         MVI   LETTER,Y
         WTO   MESSAGE
         MVI   LETTER,Z
         WTO   MESSAGE
         RETURN
         DS    0CL24
         DC    CL22'DATA5A ... Letter is <'
LETTER   DC    CL1' '
         DC    CL1'>'
W        EQU   C'A'
X        EQU   X'C1'
Y        EQU   B'11000001'
Z        EQU   193
         END   BEGIN
A:\MIN>m370 data5a
A:\MIN>a370 data5a/lx
A:\MIN>l370 data5a/lx
A:\MIN>data5a
DATA5A ... Letter  is <A>
DATA5A ... Letter  is <A>
DATA5A ... Letter  is <A>
DATA5A ... Letter  is <A>
```

5.4. Aritmética binaria

(miramos Además, sólo) es realmente muy simple. Es muy similar a la suma decimal, excepto que usted tiene que recordar que, igual que "llevar a la" cada vez que llegue a las diez de decimales Además, se "llevan una" cada vez que llegue a las dos de la adicción binaria. Por ejemplo:

```
1. 11010
               + 1011
                        1 0 + 1 = 1 -> no desbordamiento
                     1
2. 11010
              + 1011
                       01 1 + 1 = 2 -> 2 no es un dígito válido,
                                           introduzca 0 y llevar el 1.
3. 11010
               + 1011
                     101 1 + 0 + 0 = 1 -> no desbordamiento                  1 1
4. 11010
               + 1011
                    0101 1 + 1 = 2 -> escriba 0 y llevar el 1.                1 1
5. 11010
               + 1011
                 100101 1 + 1 = 2 -> escriba 0 y llevar el 1.
```

La respuesta es 11010 2 + 1011 2 = 100101 2.

Si usted convertir esos valores en decimal, verá este es equivalente a 26+11 = 37.

Usted probarlo ...

```
17. 10101 2 + 1101 2 = ____________ 2
18. 11011 2 + 11011 2 = ____________ 2
19. 10011011 2 + 11110001 2 = ____________ 2
```

5.5. Aritmética hexadecimal

Aritmética hexadecimal (además sólo) no es tan fácil como la aritmética binaria, pero no es tan malo, una vez uno se acostumbra a ella. Habrá ocasiones en que es necesario utilizar la aritmética hexadecimal en BAL, como cuando la depuración de su programa. Durante ese tiempo es probable que desee utilizar un calculadora. Sin embargo, una cierta comprensión de la aritmética simple hexadecimal mejorará su eficacia como un programador en ensamblador.

Como se puede imaginar, con la adición hexadecimal se "llevar a la" cada vez que llegue a dieciséis años. Considere el siguiente ejemplo:

```
1. ADE
               + C7
                      5 E + 7 = 14 + 7 = 21 = 16 + 5                    ent
2. ADE                + C7
                     A5 1 + D + C = 1 + 13 + 12 = 26 = 16 + 10
3. ADE                + C7
                   BA5 1 + A = 1 + 10sx = 11
                           entrar B (11)
```

La respuesta es $ADE_{16} + C7_{16} = BA5_{16}$. Si usted convertir esos valores en decimal, verá este es equivalente a $2782 + 199 = 2981$.

Usted probarlo ...

```
20. 28916 + 3A5 16 = ______ 16
21. 4A816 + DD7 16 = ______ 16
22. 32FE 16 + 722 16 = ______ 16
```

5.6. Ejercicios

1. Verdadero o falso.

   a) El sistema de numeración en base 2 es también conocido como el sistema de números binomiales. F T

   b) El sistema de numeración en base 2 tiene dos dígitos: 1 y 2. F T

   c) 1011 es una base válida 2, base 10, y la base de número de 16. F T

   d) Un subíndice puede ser utilizado para indicar la base de un número cuando la base está F T claro.

   e) El equivalente binario de la letra R, que es 11011001 cuando se utiliza el EBCDIC F T la secuencia de ordenación.

   f) Un poco consta de ocho bytes. F T

   g) El equivalente hexadecimal del número decimal 13 es C. F T

   h) Un byte tiene 255 valores posibles. F T

   i) 26 = 64. F T

   j) 20 = 16 0. F T

   k) 21 = 16 1. F T

   l) 12 = 116. F T

   m) 11101111 2 = EF 16. F T

2. Encontrar el equivalente binario de 11, 23, 26 y 30. ¿Qué puede concluir acerca de los binarios representación de cualquier número decimal extraño? Ni siquiera un número decimal?

3. Complete el siguiente:

   a) 10111 2 = 10 = _____ _____ 16
   b) 11101111 2 = 10 = _____ _____ 16
   c) _____ 2 = 31 10 = _____ 16
   d) _____ 2 = 240 10 = _____ 16
   e) 2 = _____ _____ 10 = 8316
   f) 2 = _____ _____ 10 = AE16

4. Complete el siguiente:

   a) 11101 2 = 10 = _____ _____ 16
   b) 10101011 2 = 10 = _____ _____ 16
   c) _____ 2 = 48 10 = _____ 16
   d) _____ 2 = 221 10 = _____ 16
   e) 2 = _____ _____ 10 = 7F16
   f) 2 = _____ _____ 10 = BC16

5. Complete el siguiente:

   a) 11010 2 = 10 = _____ _____ 16
   b) 10100001 2 = 10 = _____ _____ 16
   c) _____ 2 = 67 10 = _____ 16
   d) _____ 2 = 234 10 = _____ 16
   e) 2 = _____ _____ 10 = ED16
   f) 2 = _____ _____ 10 = 6A16

6. 6. Realice la siguiente base 2 adiciones:

   a) a.
   b) b. 10110011 10001111 + 111001 + 1011101
   c) c.
   d) d. 10011100 11010001 + 10101110 + 11101101
   e) Encontrar el equivalente decimal a cada una de las sumas en las partes de anuncios. e.

7. Realizar la base de los siguientes 16 adiciones:

a) a.

b) b. CA18 7EFA + 797 + 322

c) c.

d) d. 2DB2 DEAD + 693 + ACE

e) Encontrar el equivalente decimal a cada una de las sumas en las partes de anuncios. e.

8. Este capítulo incluye una tabla que muestra la representación EBCDIC de todas las letras mayúsculas. Tome el número de decimales para las letras A, B, R, S y Z, y encontrar su equivalente binario. Los números decimales para los equivalentes en minúsculas de estas cartas son 129, 130, 153, 162, y 169, respectivamente. Encuentra los equivalentes binarios de estos números también. Utilice estos números binarios para responder a la siguiente pregunta: En EBCDIC, ¿cuál es la diferencia entre una letra mayúscula y su equivalente minúscula?

9. Este capítulo incluye una tabla que muestra la representación EBCDIC de todos los números, del 0 al 9 inclusive. Buscar el equivalente hexadecimal para cada uno de estos números. Utilice estos números hexadecimales para responder a la siguiente pregunta: En EBCDIC, ¿qué tienen en común todos los números?

Capítulo 6

¿Qué es esta cosa de la izquierda?

Objetivos Al término de este capítulo usted será capaz de:

- Describir el formato de instrucción SI tal como se utiliza con la MVI y CLI instrucciones,
- Describir el formato de instrucción de las SS como se utiliza con el MVC y la CVX instrucciones,
- Describir el formato de instrucción RX tal como se utiliza con la instrucción antes de Cristo,
- Dado un parcial. PRN con el código fuente, determinar el código de objeto perdido, y
- Dado un parcial. PRN con el código objeto, determinar el código fuente que falta.

Introducción Ahora vamos a dedicar algún tiempo a la lista de montaje en detalle. Para PC/370, esta es la . PRN archivo producido por A370. Nuestro propósito al hacerlo es conseguir una mejor comprensión de la proceso de montaje, y para aprender acerca de los tipos de instrucciones. Este conocimiento es importante para propósitos de depuración. Incluso aprenderá a "invertir el montaje", un programa, es decir, teniendo en cuenta algunos código objeto, determinar las instrucciones de origen.

* * * * * * * * * * * * * * * * * * * *

Cada instrucción tiene un tipo de instrucción. Hasta ahora, hemos visto a las cinco instrucciones en detalle: MVI, MVC, CLI, CLC, y BC. (Recuerde que todas las instrucciones de varias sucursales son variaciones de la BC la instrucción.) macros, como la OMC y GET, constará de múltiples instrucciones: no se discutidos aquí. Considere lo siguiente (sin duda sin sentido) del programa, STUFF6A.MLC, que utiliza estas instrucciones:

```
STUFF6A BEGIN
        CLC     A,B
        BL      D
        MVC     C,B
        MVI     E,C'1'
        B       F
D       EQU     *
        CLI     B,C'D'
        BE      F
        MVC     C,A
        MVI     E,C'9'
F       EQU     *
        RETURN
A       DC      CL3'AB'
B       DC      CL4'CDEF'
C       DS      CL2
E       DC      CL1'*'
        END
```

La página siguiente muestra la asamblea (. PRN) el anuncio de este programa. Las porciones de las programa que queremos concentrarnos en la sombra han sido (dejamos de lado la expansión de macros.)

```
STUFF6A                                                  PAGE    1
PC/370 CROSS ASSEMBLER   OPTIONS=LXACE
   LOC                  ADR1   ADR2 LINE  LABEL    OP        OPERANDS
000000                                 1 *******  BEGIN
000000                                 2 STUFF6A  CSECT
000000                                 3          USING     *,15
000000 47F0F058                0058    4          B         KZHQX001
000004 0B                              5          DC        AL1(11)
000005 E2E3E4C6C6F6C140                6          DC        CL11'STUFF6A '
000010 0000000000000000                7 HZQKX001 DC        18F'0'
000058 90ECD00C                000C    8 KZHQX001 STM       14,12,12(13)
00005C 50D0F014                0014    9          ST        13,HZQKX001+4
000060 18ED                           10          LR        14,13
000062 41D0F010                0010   11          LA        13,HZQKX001
000066 50D0E008                0008   12          ST        13,8(0,14)
00006A                                13          DROP      15
00006A                                14          USING     HZQKX001,13
00006A D502D08ED091    009E    00A1   15          CLC       A,B
000070 4740D072                0082   16          BL        D
000074 D201D095D091    00A5    00A1   17          MVC       C,B
00007A 92F1D097                00A7   18          MVI       E,C'1'
00007E 47F0D084                0094   19          B         F
```

```
000082              00000082          20  D        EQU       *
000082 95C4D091                 00A1  21           CLI       B,C'D'
000086 4780D084                 0094  22           BE        F
00008A D201D095D08E    00A5     009E  23           MVC       C,A 000090 92F9D097
000094              00000094          25  F        EQU       *
000094                                26  *+++++++ RETURN
000094 58DD0004                 0004  27           L         13,4(13)
000098 98ECD00C                 000C  28           LM        14,12,12(13)
00009C 07FE                           29           BR        14
00009E C1C240                         30  A        DC        CL3'AB'
0000A1 C3C4C5C6                       31  B        DC        CL4'CDEF'
0000A5                                32  C        DS        CL2
0000A7 5C                             33  E        DC        CL1'*'
0000A8                                34           END
```

La MVI y las instrucciones de la CLI son de tipo instrucciones SI, el MVC y las instrucciones de la CVX son de tipo SS instrucciones, y la instrucción antes de Cristo es una instrucción de RX tipo. Estos tipos de instrucción son comúnmente ilustra de la siguiente manera:

```
+---------------+-------------+-------------+
                  | 1 de halfword | 2 halfword | 3 halfword |
          +------+-------+-------+---+----------+----------- --- +
          | SI | Op. Código | I 2 | B1 | D1 |
          +------+-------+-------+---+----------+---+------- --- +
          | SS * | Op. Código | L | B 1 | D1 | B 2 | D2 |
          +------+-------+---+---+---+----------+---+------- --- +
          | RX | Op. Código | R 1 | X2 | B2 | D2 |
          +------+-------+---+---+---+----------+
```

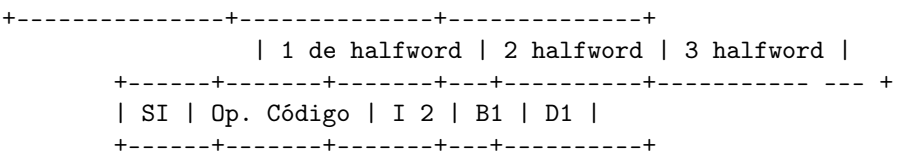
*Hay más de una forma de instrucción de las SS. Esta es la veremos por ahora ...

donde:

```
Código de Operación de
                Op. Código. .                              Base de registro de denc
                B1, B2. .                                Campo de desplazamiento
                D1, D2. .                                Campo operando inmediato
                I2. . . .                                Campo Longitud
                L. . . .                                Indice de campo designación R
                X2. . . .
```

Tenga en cuenta que todas las instrucciones de uso de uno, dos, o tres medias palabras. Recuerde que una fullword es de cuatro octetos, por lo que una media palabra es de dos bytes. Por lo tanto, todas las instrucciones ocupan dos, cuatro, o seis bytes.

6.1. El formato de instrucción SI



Todas las instrucciones que utilizan el SI, o el almacenamiento inmediato, formato de instrucción de ocupar dos medias palabras, o cuatro bytes, como se muestra en la ilustración anterior.

El primero de estos cuatro bytes es el código de operación, o Op. Código. Cada instrucción ensamblador es le asigna un valor decimal de 0 a 255, tal como lo son las letras y números. Una instrucción puede incluso tienen el valor decimal igual que una letra o un número! Por ejemplo, el CLI es Op. Código X'95 ', que es el mismo que n una minúscula ', y la MVI es Op. Código X'92', que es el mismo k minúscula '. La CPU puede decir la diferencia por el contexto en el que se encuentra. Mira nuestra muestra programa, una parte de la cual se repite aquí. Usted puede ver los códigos de Op. de 95 y 92.

```
00006A D502D08ED091 009E 00A1 15 CLC A, B,
000070 4740D072 0082 16 BL D
000074 D201D095D091 00A5 00A1 17 MVC C, B,
00007A 92F1D097 00A7 18 MVI E, C'1 '
00007E 47F0D084 0094 19 BF
000082 00000082 20 D EQU *
000082 95C4D091 00A1 21 CLI B, C'D '
000086 4780D084 0094 22 BE F
00008a D201D095D08E 00A5 009E 23 MVC C, A
000090 92F9D097 00A7 24 MVI E, C'9 '
000094 00000094 25 F EQU *
```

El segundo byte es el valor de inmediato. En el caso de la MVI y las instrucciones de la CLI, el valor que ser movido o comparación es en realidad una parte de la instrucción! Por ejemplo, en la línea 18, que queremos mover un '1 'al campo E. El EBCDIC equaivalent a un '1' es el decimal 241, o F1 hexadecimal, y se puede ver de que la F1 es parte de la instrucción, inmediatamente después del código de operación de 92 años. Del mismo modo, en la línea 21, queremos comprobar si el primer byte del campo B es la letra D. El equivalente EBCDIC a una 'D' es 196 decimal, hexadecimal o C4, y se puede ver que el C4 es parte de la instrucción, inmediatamente después de la el código de operación de 95.

La siguiente parte, la base y el desplazamiento, es un poco más confuso, pero es algo que debe entender si va a tener éxito con el lenguaje ensamblador. La media palabra siguiente de la la instrucción indica la dirección del primer operando, el campo de destino o en comparación con el. Dentro de estos dos bytes, el primer byte (que es sólo cuatro bits, o un dígito hexadecimal) indica

que la base de registro. Recuerde que en nuestro análisis de los números binarios de cuatro bits que podría oscilar en el valor de 0 (0000, o todos los bits apagados) a 15 (1111, o en todos los bits). Esta es también la gama de registros: tenemos dieciséis registros, numerados del 0 al 15 inclusive. Así que estos cuatro bits designar a un registro. En todos los tres de las instrucciones de arriba (líneas 18, 21 y 24), el próximo medio byte (a raíz de la inmediata valor) es D, que es el equivalente en hexadecimal a decimal 13. Esto nos dice que el registro 13 se apunta a la operandos, o cerca de ellos. ¿Cómo puede ser esto? ¿Dónde nos lo inicializar registro de 13? No lo hicimos, pero los usos BEGIN macro de registro 13, como el registro de base. Esto se hace mediante la instrucción USING en la línea 14, que es parte de la expansión de la macro BEGIN. Registro de 13 puntos a HZQKX001 (a etiqueta sin sentido generado por la macro BEGIN), que es el comienzo del programa. El siguiente y uno de bytes de la mitad (que es doce bits) indica el desplazamiento, o la distancia, del operando de HZQKX001.

En la línea 18, el desplazamiento es X'097. La dirección de HZQKX001 es X'000010 '(en la columna de conciencia a la línea 7 del programa). Así que si añadimos X'097 a X'000010 'tenemos X'0000A7', que es la LOC para E campo!

En la línea 21, el desplazamiento es X'091. La dirección de HZQKX001 es X'000010 '(en la columna de conciencia a la línea 7 del programa). Así que si añadimos X'091 a X'000010 'tenemos X'0000A1', que es la LOC para B campo!

El ensamblador nos ahorrará algo de este trabajo. En su caso al ensamblador colocará el la dirección de los operandos en las columnas ADR1 o ADR2. Por ejemplo, en la línea 18, en donde queremos avanzar "1" al campo E, ADR2 es 00A7, lo mismo que hemos encontrado, añadiendo el desplazamiento a la base. Asimismo, en línea 21, donde se quiere comprobar si el primer byte del campo B es la letra D, ADR2 es 00A1, lo mismo que hemos encontrado anteriormente.

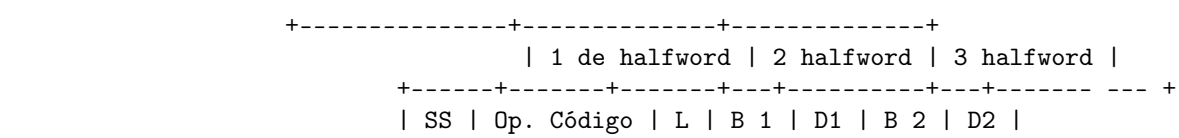
Es evidente que las columnas ADR1 y ADR2 son más fáciles de usar. Entonces, ¿por aprender la otra manera? Porque cuando estamos mirando el código objeto solamente, como por ejemplo durante una sesión de pruebas o cuando la depuración, la ADR1 y la Columnas ADR2 no estará disponible.

Usted probarlo ... 1. Mostrar el código objeto para la línea 18 si esa línea se cambió a la MVI E, C'2 '

2. Mostrar el código objeto para la línea 21, si esa línea se ha cambiado a la CLI B 1, C '$'

3. Mostrar el código objeto para la línea 24 si esa línea se cambió a la MVI E +1, X'40 '

6.2. El formato de instrucción SS



```
+------+-------+-------+---+----------+---+------- --- +
```

Todas las instrucciones que utilizan las SS, o de almacenamiento a almacenamiento, formato de instrucción de ocupar tres medias palabras, o seis bytes, como se muestra en la ilustración anterior.

Como con todas las instrucciones de ensamblador, el primero de los seis octetos es el código de operación, o Op. Código. CLC es Op. Código X'D5 '(lo mismo que una mayúscula' N ') y MVC es Op. Código X'D2' (lo mismo que un superior case 'K'). Mira a nuestro programa de ejemplo, una parte de la cual se repite aquí. Usted puede ver el Op. Códigos de D5 y D2.

```
00006A D502D08ED091   009E   00A1 15   CLC A,B
000070 4740D072              0082 16   BL  D
000074 D201D095D091   00A5   00A1 17   MVC C,B
00007A 92F1D097              00A7 18   MVI E,C'1'
00007E 47F0D084              0094 19   B   F
000082               00000082      20 D EQU *
000082 95C4D091              00A1 21   CLI B,C'D'
000086 4780D084              0094 22   BE  F
00008A D201D095D08E   00A5   009E 23   MVC C,A
000090 92F9D097              00A7 24   MVI E,C'9'
000094               00000094      25 F EQU *
```

El segundo byte es la longitud de la operación. Recuerde que un solo byte puede variar en el valor de 0 a 255. Pero un movimiento de la longitud de 0 no tiene ningún sentido. Así que la longitud de operando es en realidad la la duración de la operación menos uno. De esta manera, los valores de 0 a 255 indican longitudes de operación de (se mueve, compara, etc), de 1 a 256 inclusive. Esta es la razón por la MVC y la CVX instrucciones son limitado a 256 caracteres.

En la línea 15, se compara un campo con el campo B. El campo A se define como 3 bytes de longitud, mientras que el campo B es definida como 4 bytes de longitud. Recuerde que en nuestro análisis de la instrucción de CLC que la longitud de la comparar es determinada por la longitud del primer operando. Desde ese operando, un campo, es de 3 octetos de largo, 3 bytes serán comparados. Esto se muestra en la instrucción como 3 menos uno, o 2. El hexágono representación de los 2 es X'02 ', y eso es lo que sigue el código de operación, es decir, un Convenio de Responsabilidad Civil por un período de 3 es representado como X'D502.

En la línea 17, pasamos al campo de campo B C. Campo B se define como 4 bytes de longitud, mientras que el campo C se define como 2 bytes de longitud. Sabemos que la longitud de un MVC está determinada por la longitud de la primera operando. Desde ese operando, Campo C, es de 2 bytes de longitud, 2 bytes serán comparados. Esto se muestra en la instrucción como 2 menos uno, o 1. La representación hexadecimal de 1 es X'01 ', y eso es lo que sigue, el código de operación, es decir, un MVC para una longitud de 2 se representa como X'D201.

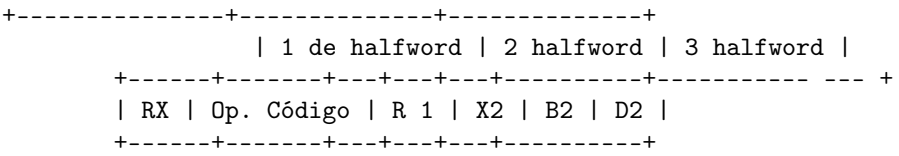
Bytes El halfword segundo (tercero y cuarto) indica que la base y el desplazamiento de la primera octetos operando, y la media palabra tercero (quinto y sexto) indica que la base y el desplazamiento de la segundo operando. Esto funciona de la misma como se describe en la instrucción de formato SI anterior, excepto que hay dos operandos en lugar de uno. Nuestro análisis anterior de las columnas ADR1 y ADR2 se aplica también.

Usted probarlo ... 4. Mostrar el código objeto para la línea 15 si esa línea se cambió a la CVX, B, A

5. Mostrar el código objeto para la línea 15 si esa línea se cambió a la CVX A 1 (2), B 2

6. Mostrar el código objeto para la línea 23, si esa línea se cambió a MVC C (3), B

6.3. El formato de instrucción RX



Todas las instrucciones que utilizan la RX, o de registro de índice, formato de instrucción ocupan dos medias palabras, o de cuatro bytes, como se muestra en la ilustración anterior.

Una vez más, la primera de estas cuatro bytes es el código de operación. BC es Op. Código X'47 '. Mira nuestra programa de ejemplo: se puede ver el código de operación de 47.

```
00006A D502D08ED091 009E 00A1 15 CLC A, B,
000070 4740D072 0082 16 BL D
000074 D201D095D091 00A5 00A1 17 MVC C, B,
00007A 92F1D097 00A7 18 MVI E, C'1 '
00007E 47F0D084 0094 19 BF
000082 00000082 20 D EQU *
000082 95C4D091 00A1 21 CLI B, C'D '
000086 4780D084 0094 22 BE F
00008a D201D095D08E 00A5 009E 23 MVC C, A
000090 92F9D097 00A7 24 MVI E, C'9 '
000094 00000094 25 F EQU *
```

El próximo medio byte (cuatro bits) corresponde a la máscara. El valor de esta máscara, que va desde 0000 (todos los bits apagados) a 1111 (en todos los bits), está determinado por la regla mnemotécnica (BE, BL, etc) y corresponde a los valores posibles del código de condición que hemos discutido en un capítulo

anterior. Por ejemplo, nos dijo que si se comparan dos campos, A y B, y A es inferior a B, el resultante código de condición sería 01002, o X'4. En la línea 16 anterior, queremos poder a D, si A es menos de B. Así que la máscara es X'4 ', que aparece inmediatamente después del código de operación de X'47'.

El próximo medio byte indica el registro de índice. Un registro de índice es como un desplazamiento cuyo valor depende de un registro. Considerando que los otros desplazamientos vimos fueron constante, el valor del registro, y por lo tanto el desplazamiento total, pueden variar. Cuando el índice de registro es el 0, se ignora. Esto suele ser el caso. En todas las ramas de arriba, el índice de registro es cero.

Hemos visto que un BL se muestra como X'4740 ', una B (incondicional) como X'47F0', y un BE como X'4780 '. Para completar, se muestra el siguiente cuadro:

```
BE X'4780 '
 BH X'4720 '
 BL X'4740 '
BNE X'4770 '
BNH X'47D0 '
BNL X'47B0 '
 B X'47F0 '
```

Como antes, la palabra la segunda mitad de la enseñanza da la base y el desplazamiento de la etiqueta para que la instrucción debe rama. La única diferencia entre este y el SI y SS formatos de instrucción que hemos discutido es que, una vez más, hay un registro de índice que permite para un segundo nivel de desplazamiento.

Usted probarlo ... 7. Mostrar el código del objeto en la línea 16 si esa línea se cambió a BH D

8. Mostrar el código objeto para la línea 19 si esa línea se cambió a BD

9. Mostrar el código objeto para la línea 22 si esa línea se ha cambiado a BNE F

6.4. Instrucciones de Definición de Datos

Por último, dirigimos nuestra atención a las instrucciones de definición de datos, DS y DC.

```
00009E C1C240 30 A CL3'AB DC
0000A1 C3C4C5C6 31 B DC CL4'CDEF '
0000A5 32 C DS CL2
0000A7 5C 33 E DC CL1 '*'
```

Como se puede ver, cuando una instrucción de DS se utiliza, no se genera código objeto: el lado izquierdo de la lista está en blanco. Sin embargo, la ubicación

del contador se incrementa. Por ejemplo, el campo comienza en la C ubicación 0000A5, sobre una longitud de dos. Por lo tanto, el siguiente campo (E) se iniciará en la localidad 0000A5 + 0002 = 0000A7.

Cuando se utiliza una instrucción de CC, se muestra la representación hexadecimal de los datos. Si el campo es más de ocho bytes de longitud, el primero de ocho bytes (sólo) se muestran. Al igual que con la instrucción de DS, la contador de ubicación se incrementa. Por ejemplo, un campo de comenzar en la ubicación 00009E, para una longitud de tres. Por lo tanto, el siguiente campo comenzará a las 00009E ubicación + 0003 = 0000A1. El campo A se inicializa a C'ABb ', que se muestra como X'C1C240. X'40 'es un espacio en blanco: memorizar eso!

Usted probarlo ... 10. Mostrar el código objeto para la línea 30, si esa línea se cambió a

```
A CL3'A DC
```

11. Mostrar el código objeto para la línea 31 si esa línea se ha cambiado a

```
B DC CL4'12 '
```

12. Mostrar el código objeto para la línea 33 si esa línea se cambió a

```
E CC 2X'FE '
```

6.5. Ejercicios

1. Verdadero o falso.

   a) La MVI MVC y las instrucciones son de tipo SS. F T

   b) Cada instrucción ocupa exactamente una, dos, o tres medias palabras. F T

   c) El segundo byte de una instrucción MVI contiene la longitud de la medida, menos F T uno.

   d) Registro 13 es nuestro registro de base, como se indica en la macro BEGIN. F T

   e) Si el primer operando de una MVC se indica por X'D010 ', entonces el objetivo de F T (recepción) de campo comienza diez bytes pasado donde registro 13 está señalando.

   f) Si la longitud de la CLC se indica por X'0C ', y luego doce bytes se están F T comparación.

   g) El código de operación de BE y BNE son X'47 '. F T

   h) Habida cuenta de E DC C'B ', la materia de la izquierda se X'C2. F T

i) Dado F DC CL2'B ', la materia de la izquierda se X'C2C2. F T

j) X'4708 'es el código objeto de BC con una máscara de 8. F T

k) Dado A CL3 DS en LOC = 0000AE, entonces A 2 es LOC = 0000B0. F T

l) Dado R DS CL1, las cosas de la izquierda se X'D9. F T

m) El designador de registro base (B1 o B2) ocupa un byte de código objeto. F T

2. Teniendo en cuenta generados en

```
92C2D08A LOC = 0000BC, y D se define como CL3.
MVI D, C'B '
```

a) ¿Cuál es la LOC para la siguiente instrucción?

b) Mostrar el código objeto si esta línea se han cambiado a la

```
MVI D, C'C '
```

c) Mostrar el código objeto si esta línea se han cambiado a la

```
MVI D +1, C'D '
```

d) Mostrar el código objeto si esta línea se han cambiado a la

```
MVI D +2, x'DD '
```

e) Mostrar el código objeto si esta línea se ha cambiado a la

```
CLI D, C'E '
```

f) Mostrar el código objeto si esta línea se ha cambiado a la

```
CLI D +1, B'11100010 '
```

3. Teniendo en cuenta

```
D20AD090D080 generados en LOC = 0000D6, e Y se define como el CL14.
MVC X, Y
```

a) ¿Cuánto tiempo es X?

b) ¿Cuál es la LOC para la siguiente instrucción?

c) Mostrar el código objeto si esta línea se ha cambiado a

```
MVC X (4), Y
```

d) Mostrar el código objeto si esta línea se ha cambiado a

```
MVC X 2 (5), Y
```

e) Mostrar el código objeto si esta línea se ha cambiado a

```
MVC X +3 (1), Y 4
```

f) Mostrar el código objeto si esta línea se ha cambiado a

```
MVC Y, X
```

g) Mostrar el código objeto si esta línea se han cambiado a la

```
CVX, X, Y
```

h) Mostrar el código objeto si esta línea se han cambiado a la

```
CVX, Y (3), X +1
```

4. Ejercicios 4.

a) Llene los espacios en blanco (líneas 15-21 y 27):

```
STUFF6B                                                  PAGE     1
PC/370 CROSS ASSEMBLER OPTIONS=LXACE
   LOC                ADR1    ADR2 LINE LABEL    OP        OPERANDS
000000                                1 *+++++++  BEGIN
000000                                2 STUFF6B  CSECT
000000                                3          USING     *,15
000000 47F0F058               0058    4          B         KZHQX001
000004 0B                             5          DC        AL1(11)
000005 E2E3E4C6C6F6C240               6          DC        CL11'STUFF6B '
000010 0000000000000000               7 HZQKX001 DC        18F'0'
000058 90ECD00C               000C    8 KZHQX001 STM       14,12,12(13)
00005C 50D0F014               0014    9          ST        13,HZQKX001+4
000060 18ED                          10          LR        14,13
000062 41D0F010               0010   11          LA        13,HZQKX001
000066 50D0E008               0008   12          ST        13,8(0,14)
00006A                               13          DROP      15
00006A                               14          USING     HZQKX001,13
00006A D201D086D088   0096    0098   15          MVC       ____,Y
000070 92F9D08D               009D   16          MVI       Z,____
000074 95D3D086               0096   17          CLI       ____,C'L'
000078 4780D07C               008C   18          ____      W
00007C D501D086D08D   0096    009D   19          ____      X,Z
000082 4740D07C               008C   20          BL        ____
000086 D203D08DD088   009D    0098   21          ____      ____,Y
00008C              0000008C         22 W        EQU       *
00008C                               23 *++++++++ RETURN
00008C 58DD0004               0004   24          L         13,4(13)
000090 98ECD00C               000C   25          LM        14,12,12(13)
000094 07FE                          26          BR        14
000096 D1D2                          27 X        DC        CL2'____'
000098 D3D4D5D6D7                    28 Y        DC        CL5'LMNOP'
00009D D8D9E2E3                      29 Z        DC        CL4'QRST'
0000A8                               30          END
```

b) Usted es el equipo ... ¿cuáles son los valores de X, Y, Z, después de este programa de corre?

X = _____ Y =_____ Z =_____

5. Ejercicios 5.

a) Llene los espacios en blanco (líneas 15-19 y 25-27):

```
STUFF6C                                                 PAGE     1
PC/370 CROSS ASSEMBLER OPTIONS=LXACE
   LOC                 ADR1    ADR2 LINE LABEL    OP        OPERANDS
000000                                1 *+++++++  BEGIN
000000                                2 STUFF6C   CSECT
000000                                3           USING     *,15
000000 47F0F058               0058    4           B         KZHQX001
000004 0B                             5           DC        AL1(11)
000005 E2E3E4C6C6F6C340               6           DC        CL11'STUFF6C '
000010 0000000000000000               7 HZQKX001  DC        18F'0'
000058 90ECD00C               000C    8 KZHQX001  STM       14,12,12(13)
00005C 50D0F014               0014    9           ST        13,HZQKX001+4
000060 18ED                          10           LR        14,13
000062 41D0F010               0010   11           LA        13,HZQKX001
000066 50D0E008               0008   12           ST        13,8(0,14)
00006A                               13           DROP      15
00006A                               14           USING     HZQKX001,13
00006A ____D07CD07A   008C    008A   15           CLC       D,F
000070 ____D06C               007C   16           BL        H
000074 ____D07D               008D   17           MVI       E,C'D'
000078 ____D070               0080   18           B         G
00007C ____D07C               008C   19 H         MVI       D,C'F'
000080              00000080         20 G         EQU       *
000080                               21 *++++++++ RETURN
000080 58DD0004               0004   22           L         13,4(13)
000084 98ECD00C               000C   23           LM        14,12,12(13)
000088 07FE                          24           BR        14
00008A C4C5                          25 F         DC        __________
00008C C4                            26 D         DC        __________
00008D C6C5C4                        27 E         DC        __________
000090                               28           END
```

b) Usted es el equipo ... ¿cuáles son los valores de F, D y E después de este programa corre?

F = _____ D =_____ E =_____

6. Ejercicios 6.

a) Llene los espacios en blanco (líneas 15-19):

```
STUFF6D                                                 PAGE    1
PC/370 CROSS ASSEMBLER OPTIONS=LXACE
  LOC                ADR1   ADR2 LINE LABEL    OP        OPERANDS
000000                                1 *******  BEGIN
000000                                2 STUFF6D  CSECT
000000                                3          USING     *,15
000000 47F0F058              0058     4          B         KZHQX001
000004 0B                             5          DC        AL1(11)
000005 E2E3E4C6C6F6C440               6          DC        CL11'STUFF6D '
000010 0000000000000000               7 HZQKX001 DC        18F'0'
000058 90ECD00C              000C     8 KZHQX001 STM       14,12,12(13)
00005C 50D0F014              0014     9          ST        13,HZQKX001+4
000060 18ED                          10          LR        14,13
000062 41D0F010              0010    11          LA        13,HZQKX001
000066 50D0E008              0008    12          ST        13,8(0,14)
00006A                               13          DROP      15
00006A                               14          USING     HZQKX001,13
______ ________              008D    15          MVI       M,C'L'
______ ____________   008A   008D    16          MVC       L(2),M
______ ________              008A    17          CLI       L,C'N'
______ ________              0080    18          BNE       N
______ ________              008A    19          MVI       L,C'M'
000080              00000080         20 N        EQU       *
000080                               21 ******** RETURN
000080 58DD0004              0004    22          L         13,4(13)
000084 98ECD00C              000C    23          LM        14,12,12(13)
000088 07FE                          24          BR        14
00008A D4D6D4                        25 L        DC        CL3'MOM'
00008D 4040                          26 M        DC        CL2' '
000090                               27          END
```

b) Usted es el equipo ... ¿cuáles son los valores de L y M después de este programa corre?

L= ________ M=________

7. Ejercicios 7.

a) Fill in the blanks (lines 15-19 and 25-27):

```
STUFF6E                                                   PAGE    1
PC/370 CROSS ASSEMBLER OPTIONS=LXACE
   LOC                    ADR1     ADR2 LINE   LABEL    OP        OPERANDS
000000                                     1  *******  BEGIN
000000                                     2  STUFF6E  CSECT
```

```
000000                                          3           USING     *,15
000000 47F0F058                    0058         4           B         KZHQX001
000004 0B                                       5           DC        AL1(11)
000005 E2E3E4C6C6F6C540                         6           DC        CL11'STUFF6E
000010 0000000000000000                         7  HZQKX001 DC        18F'0'
000058 90ECD00C                    000C         8  KZHQX001 STM       14,12,12(13)
00005C 50D0F014                    0014         9           ST        13,HZQKX001+4
000060 18ED                                    10           LR        14,13
000062 41D0F010                    0010        11           LA        13,HZQKX001
000066 50D0E008                    0008        12           ST        13,8(0,14)
00006A                                         13           DROP      15
00006A                                         14           USING     HZQKX001,13
00006A D501D07FD081        008F    0091        15  P        _____     __________
000070 4720D072                    0082        16           _____     __________
000074 D201D081D07C        0091    008C        17           _____     __________
00007A 92F2D081                    0091        18           _____     __________
00007E 47F0D05A                    006A        19           _____     __________
000082                  00000082               20  Q        EQU       *
000082                                         21  *******  RETURN
000082 58DD0004                    0004        22           L         13,4(13)
000086 98ECD00C                    000C        23           LM        14,12,12(13)
00008A 07FE                                    24           BR        14
00008C F1F2F3                                  25  T        _____     __________
00008F F3F3                                    26  S        _____     __________
000091 F3F3                                    27  R        _____     __________
000098                                         28           END
```

b) Usted es el equipo ... ¿cuáles son los valores de T,S y R después de este programa corre?

T= _____ S=_____ R=_____

Capítulo 7

Bolsas de aritmética decimal

Objetivos Al término de este capítulo usted será capaz de:

- Dado un campo en hexadecimal, identificar la zona y los bits numéricos,
- Dado un número decimal por zonas, muestran su representación hexadecimal,
- Dado un número decimal por zonas, paquete de ese número y mostrar la representación hexadecimal de que colmaron el número,
- Dado un número de pic nic, desempaquetar ese número y mostrar la representación hexadecimal de que número sin envasar,
- Utilice la instrucción MVZ para eliminar el signo de un número sin envasar,
- Utilice la instrucción de AP para añadir un número de envasados a otro,
- Utilice la instrucción de SP para restar un número de pic nic de otro,
- Utilice la instrucción de ZAP para avanzar un número de pic nic a otro,
- Utilice la instrucción CP para comparar dos números empaquetados,
- Mostrar cómo la instrucción antes de Cristo se puede utilizar para detectar una condición de desbordamiento a raíz de una añadir o operación de restar, y
- Escribir un programa que requiere el uso de la aritmética decimal empaquetado y la presentación de los resultados de de que la aritmética.

Introducción Hasta ahora, ninguno de nuestros programas han utilizado la aritmética. Sólo hay dos maneras de hacer la aritmética en BAL: Bolsas de la aritmética decimal o binario (registro) de la aritmética. En este capítulo se presenta la aritmética decimal empaquetado, específicamente, la adición y sustracción. Hablaremos de la multiplicación y la división en un capítulo posterior.

Nos aleja también a la discusión de la aritmética binaria, porque necesidad de entender los números empaquetado, antes de que pueda trabajar con números binarios (un número binario debe convertirse a embalados antes de que se pueden imprimir.)

7.1. Presentación de Dindón Industrias

Para este capítulo se presenta un nuevo archivo. (Los archivos de la Escuela Superior Small Town de la Comunidad base de datos que hemos utilizado hasta este punto no contienen suficientes campos numéricos para introducir la aritmética.) Este archivo es el archivo de inventario para las Industrias Dindón. Dindón vende tres productos (Gizmos, Widgets, y Junque) en cuatro estados (California, Illinois, Utah y Wisconsin). El diseño de registro para el archivo de inventario, COGS.DAT, se muestra en la página siguiente. El archivo de datos es construye como sigue:

```
A: \> copy con cogs.dat
 Gizmos 020030020020017099023122529 99
 WIDGETS 015010010002022034019001100 25
 Junqué 025015015018030052010015503 39
 ^Z
          1 archivo (s) copied
  Campo Campo           N º Nombre Descripción Inicio Término Len Formato
2 CALIF ventas California 11 13 3 ZD
3 ILL ventas Illinois 14 16 3 ZD
4 UTAH ventas de Utah 17 19 3 ZD
5 WISC ventas de Wisconsin 20 22 3 ZD
6 COMIENZAN A partir inv. 23 25 3 ZD
7 Compras purch 26 28 3 ZD
8 QOH Cantidad a mano 29 31 3 ZD
9 Coste COSTE (cada uno) 32 35 4 99V99
10 Vendo Vendo por (cada uno) 36 39 4 99V99
11 CRLF PC/370 Sólo 40 41 2 CR / LF
A: \> dir cogs.dat
  Volumen en la unidad A no tiene etiqueta
  Directorio de A: \
 Costo de Ventas DAT 123 9-29-93 9:38 una
          1 archivo (s) 130560 bytes libres
```

En este capítulo vamos a crear dos informes. El primer informe, las ventas Recap, aparecerá como siguiente:

```
1 2 3 4 5 6          12345678901234567890123456789012345678901234567890123456789012345678901234567890
         Producto Calif malos Utah Wisconsin TOTAL
---------- ----- ----- ----- ----- -----
```

```
Gizmos 020 030 020 020 090
WIDGETS 015 010 010 002 037
Junqué 025 015 015 018 073
003 registros procesados.
```

Tenemos que saber cómo hacerlo, además, a fin de producir la columna de totales y recuento de registro.

7.2. Bolsas de Formato Decimal

Bolsas de decimales es un formato de almacenamiento único para los equipos de IBM System/370 y compatibles: usted no encontrará Bolsas de formato decimal en las computadoras, excepto cuando tales emulando una unidad central, como en PC/370.

En decimal dividido en zonas (sin embalaje) formato, los dígitos se ocupa un único byte. Por ejemplo, California, las ventas de Junqué (arriba) es C'025 ', la representación hexadecimal de los cuales es X'F0F2F5. Cada byte consta de dos partes: la zona (o signo) parte (todos los F en este caso), y la parte numérica (0, 2, y 5). Cada porción ocupa cuatro bits.

Si volvemos a escribir este número como:

```
F F F <- ZONA                    0 2 5 <- numeric ...
```

entonces podemos ver por qué la parte de la zona es a veces considerado como el byte de la mitad superior y la parte numérica es a veces considerado como el byte de la mitad inferior. Los bits de la zona de la derecha (en este caso el F por encima de la 5) determinar el signo de un número. Los otros F son redundantes. El signo se determina como sigue:

- indica el número no está firmado, F
- C indica el número es positivo, y
- D indica el número es negativo.

(Si usted sabe COBOL, usted recordará que un número sin signo corresponde a una cláusula de 9 PIC sin una S, como PIC 9 (3), mientras que un número con signo corresponde a una cláusula de 9 PIC con una S, tales como PIC S9 (3).)

Ejemplos:

- es 123, sin firma, X'F1F2F3 '
- es 456, y X'F4F5C6 '
- es -789 X'F7F8D9 '

Tenemos que entender cómo cada uno de estos parece si se imprimen, o si se muestra con la OMC:

- se imprime como 123, X'F1F2F3 '
- se imprime como 45F, porque X'C6 es C'F ', y X'F4F5C6 '
- se imprime como 78r, porque X'D9 es C'R. X'F7F8D9 '

Debería ser evidente por qué tendremos que hacer ajustes para que el signo cuando se imprime un número.

Usted probarlo ... 1. Dado A CL3'68P DC "es un número decimal en zonas. ¿Cuál es el valor numérico de A? ¿Está firmado? ¿Cuál sería la OMC, una pantalla?

2. Dado B DC X'F2C6 'es un número decimal en zonas. ¿Cuál es el valor numérico de la B? ¿Está firmado? ¿Qué sería de la OMC muestra B?

3. Dado CL3' C DC-39 '. Mostrar la representación hexadecimal de C. Mostrar por qué C no es válido número decimal en zonas.

4. Dado D DC CL3'39. Mostrar la representación hexadecimal de D. Mostrar por qué D no es válido número decimal en zonas.

Una vez más, los bits de la zona de determinar el signo más a la derecha, el F, son redundantes. Cuando un número es "envasado", estos bits de extrazona son eliminados, y los bits de la zona de la derecha (sólo) se mantienen. Para ejemplo, puede empacar X'F0F2F5 'en X'025F. Desde que se eliminan los bits de extrazona cuando un el número está lleno, el número de pic nic ocupa menos espacio, excepto en el caso de un decimal por zonas número de longitud 1 y 2, que aún ocupan la misma cantidad de espacio (1 y 2 bytes, respectivamente).

Considere los siguientes ejemplos:

- Nos puede embalar el número 1 (sin firmar), que ocupa un byte y se representa como X'F1, en un byte con un valor de X'1F.
- Nos puede embalar el número 12, que ocupa dos bytes y se representa como X'F1C2 ', en de dos bytes con un valor de X'012C '.
- Nos puede embalar el número -123, que ocupa tres bytes y se representa como X'F1F2D3, en dos bytes con un valor de X'123D.
- Nos puede embalar el número 1234 (sin firma), que ocupa cuatro bytes y se representa como X'F1F2F3F4 ', en tres bytes con un valor de X'01234F.
- Nos puede embalar el número 12.345, que ocupa cinco bytes y se representa como X'F1F2F3F4C5 ', en tres bytes con un valor de X'12345C.

```
Podemos resumir la cantidad de espacio ocupado de la siguiente manera:
                                                    requiere de un pic nic
```

Usted probarlo ... 5. Un campo decimal dividido en zonas de longitud de siete (es decir, con siete dígitos) se requiere un nic campo decimal de qué longitud?

6. Un campo decimal empaquetado de longitud tendrá siete dígitos ¿cuántos? La Instrucción PACK La instrucción de BAL para empacar un número es, apropiadamente, PACK. El código para implementar cada una de las ejemplos anteriores es el siguiente:

Para condensar el número 1, que ocupa un byte, en un byte:

- dónde Pack B, AA CC CL1'1 ' B DS PL1
- Para condensar el número 12, que ocupa dos bytes, en dos bytes: PACK B, donde A CL2'1B DC B DS PL2
- Para condensar el número -123, que ocupa tres bytes, en dos bytes: PACK B, donde A CL3'12L DC B DS PL2
- Para condensar el número 1234, que ocupa cuatro bytes, en tres bytes: PACK B, donde A CL4'1234 DC B DS PL3
- Para condensar el número 12.345, que ocupa cinco bytes, en tres bytes: PACK B, donde A CL5'1234E DC B DS PL3

El campo de tipo P especificador, como en PL3, indica un campo de pic nic. Es importante entender que la especificador de tipo suele ser * significativa sólo cuando se asigna un valor inicial a un campo con un DC. Nosotros puede embalar un número en un campo definido como de carácter (C), hexadecimal (X), o cualquier otro. Además, el longitud de campo se refiere al número de bytes utilizados para ese campo, no al número de dígitos que contiene. Así, por ejemplo, un campo CL3 puede contener un número de relleno de hasta cinco dígitos. Cuando el campo de tipo P especificación se usa con un DC, el valor inicial es siempre firmado. Por ejemplo:

tiene una representación hexadecimal de X'012C ',

- J DC PL2'12 ' También tiene una representación hexadecimal de X'012C ', y +12'
- K DC PL2 ' PL2'-12 'tiene una representación hexadecimal de X'012D.
- L DC Un número puede ser PACK ed en un campo que es demasiado pequeño para el número:

el orden de alto (a la izquierda la mayoría) de los dígitos se trunca, y ninguna advertencia o mensajes de error se dan. Así que ten cuidado! Un número también puede ser embalado en un campo que es mayor de lo necesario para mantener ese número. En este caso, el número se rellena con ceros a la izquierda (por lo general no es un problema).

** No es raro que un usuario por escrito macro para el tratamiento de un campo de manera diferente en función de su tipo, tales como aCLEAR macro*

que se moverá a cero a un campo si es tipo P, espacios en blanco a un campo si es de tipo C, o ceros binarios (X'00 ') a un campo si es de tipo X.

Por ejemplo, dada:

```
A CL4'1234 DC '
B DC PL2'0 '
C DC PL4'0 ' ...
```

las instrucciones:

```
PACK B, A          PACK C, A ...
```

producirá:

```
B = X'234F '          C = X'0001234F '
```

Usted probarlo ... Teniendo en cuenta N DC CL5'10639.

7. Dado W DC PL3'0 ', muestran la representación hexadecimal de W antes y después de PACK W, N

8. Dada X DC PL2'0 ', muestran la representación hexadecimal de X antes y después de PACK X, N

9. Dado Y DC CL4'0 ', muestran la representación hexadecimal de Y antes y después de PACK Y, N

10. Dado Z PL3' DC-12 ', muestra la representación hexadecimal de Z, antes y después de Pack N, Z (¡cuidado!)

* *

Si usted sabe de IBM System/370 COBOL, usted puede recordar que un número de relleno es uno con un COMP-3 cláusula de imagen. Por ejemplo, dadas las definiciones de los campos siguientes,

```
05 FLDA PIC S9 (3).
05 FLDB PIC S9 (3) COMP-3.
```

es un número de pic nic, pero FLDA no lo es. La instrucción COBOL MOVE FLDA A FLDB es equivalente a FLDB la instrucción BAL PACK FLDB, FLDA ... y eso es exactamente lo que un compilador COBOL hace: convertir COBOL las instrucciones de origen a las instrucciones de nivel equivalente de máquina! (Las instrucciones de nivel de máquina , por supuesto, varían dependiendo del tipo de máquina, es decir, un paquete se genera por un IBM/370 o compatible, pero no para un PC.)

Esta es una razón por la que los programadores de COBOL que aprender ensamblador: se desmitifica el compilador COBOL. Al saber ensamblador, a ser un mejor Programador COBOL (o cualquier otro idioma).

7.3. La Instrucción AP

El AP (añadir envasados) la instrucción se utiliza para añadir un número de envasados a otro. Por ejemplo, AP A, B agrega B a A, con B sin cambios y la suma de A. Ambos operandos deben ser válidos envasados números. Por ejemplo, dada

```
Un CL4'1234 DC '          B DC CL3'567 '          C DC PL4'0 '
```

... para añadir A y B, dando C, A y B deberán ser envasados.

Al menos un campo de trabajo se requiere:

```
PACK C, A
 PACK PK2, B,
 AP C, PK2
```

dónde

```
PK2 DC PL2'0 '
```

(A menudo se definen los campos, como la PK2 arriba, como campos de trabajo. Por ejemplo, PK3 es DC PL3'0 ', PK4 es DC PL4'0, etc partir de ahora, si me refiero a un campo de PKN, entonces se debe entender que yo soy referencia a un tipo de trabajo de campo P de tamaño n. El uso de esta convención simplificar muchos de nuestros ilustraciones.)

7.4. La Instrucción ZAP

En la codificación del ejemplo anterior (añadir y B dando C) que podría haber usado utilizar dos campos de trabajo, como Un siguiente:

```
PACK PK3, A
PACK PK2, B,
PACK C = C'0 '
AP C, PK3
AP C, PK2
```

Tenga en cuenta que mover un cero a C, a continuación, añadir PK3 a C, es el mismo que si simplemente había trasladado a PK3 C. Hay una instrucción de BAL que hacer precisamente eso: la ZAP (cero y agregar relleno) la instrucción es utiliza para mover un campo lleno hasta la otra. Por ejemplo, yo podría sustituir PACK C = C'0 y AP C, PK3 con ZAP C, PK3. También podría sustituir PACK C = C'0 con ZAP C = P'0 '.

Usted probarlo ... 11. Dada X DC CL2 ', muestra la representación hexadecimal de X después de PACK X = CL1'1'.

12. Sustituya el paquete de (11) con un ZAP.

13. Dado W DC PL2'10 ', DC PL2' X-6', Y DC CL2'15 ', y Z DC PL2'8.
Escriba el código de BAL de
determinar

```
Z = W + X + Y.
```

Mostrar la representación hexadecimal de Z después de cada instrucción. Su final respuesta de Z debe X'019C '.

(Recordatorio: Y deben acondicionarse antes de que se puede añadir. Definir un campo de trabajo si es necesario.)

7.5. La detección de una condición de desbordamiento

Al agregar (con AP) o en movimiento (con ZAP), si la suma no caben en el campo de recepción, el alto dígitos de orden se truncan. El ensamblador dará ningún mensaje de advertencia de que ello ocurra (como que la mayoría de los compiladores de COBOL), ni el programa de abend en tiempo de ejecución. La detección de tal "desbordamiento" en tiempo de ejecución es, sin embargo, un proceso muy simple. Recuerde que en nuestro estudio de IFS en BAL de que el código de condición tiene cuatro bits. El bit cuarto se usa generalmente para indicar un desbordamiento condición:

```
IGUALDAD DE BAJA ALTA OVERFLOW                                  8 4 2 1
```

Puede buscar una condición de desbordamiento mediante el uso de una sucursal, a condición de (BC) con una máscara de 1, o el BO mnemónico extendido (rama de desbordamiento). Por el contrario, puede utilizar el BNO mnemónico (rama en ninguna de desbordamiento). Esto puede demostrarse con el siguiente programa:

```
OVERFLOW BEGIN
     AP     A,B
     BO     OVER
     WTO    'There was NOT an overflow'
     B      DONE
OVER EQU    *
     WTO    'There WAS an overflow'
DONE EQU    *
     RETURN
A    DC     PL2'998'
B    DC     PL2'2'
     END    OVERFLOW
```

En este caso, usted verá el mensaje había un exceso, ya que 998 2 = 1000, y 1000 no se caben en un campo de dos bytes. Si cambia el valor de B 2 a 1, podrás ver el mensaje No se NO es un desbordamiento, ya que 998 +1 = 999, y 999 caben en un campo de dos bytes.

7.6. La Instrucción UNPK

Después de la adición anterior (adición de 1.234 a 567), C contendrá 1801, representado como X'0001801C. Tenga en cuenta que el resultado de cualquier operación aritmética (incluyendo ZAP) de la firma (con C para los positivos, y de D para el negativo), aunque los números originales fueron sin firmar! (PACK UNPK y no se consideran las operaciones aritméticas.)

Por supuesto, no puedo imprimir campo C en este formato. El primer número debe ser descomprimido. El instrucciones para hacerlo es UNPK. Esta instrucción funciona como la instrucción PACK pero a la inversa. El signo del número envasados (que es la más a la derecha cuatro bits) voy a entrar en la parte de la zona de del byte más a la derecha del número de desempaquetado. Ese campo también contendrá todos los despedidos F, que hemos hablado antes. Así, por ejemplo, UNPK WK5, C, donde WK5 DC CL5 'dará WK5 = X'F0F1F8F0C1.

(Al igual que en nuestra discusión sobre el trabajo Bolsas de campos de la PKN formulario de arriba, a menudo se definen los campos, como WK5 arriba, como campos de trabajo. Por ejemplo: WK3 DC CL3 ', WK4 DC CL4', etc De ahora en adelante, si Me refiero a un campo de WKN, entonces se entiende que me estoy refiriendo a un tipo de trabajo de campo C de tamaño de n.)

Si el campo de recepción es demasiado pequeña para el número sin envasar, los dígitos de orden superior se truncado. Por ejemplo, UNPK WK3, C dará WK3 = X'F8F0C1 '(donde WK3 se definió como WK3 DC CL3 ".) Si el campo de recepción es mayor que el necesario para mantener el número sin envasar, será alinea a la izquierda con s. X'F0 ' Así UNPK WK7, C dará WK7 = X'F0F0F0F1F8F0C1 '(donde se WK7 define como WK7 DC CL7 '.)

Usted probarlo ... Teniendo en cuenta Q PL2' DC-708 '

14. Mostrar WK3 después UNPK WK3, Q

15. Mostrar WK2 después UNPK WK2, Q Dado R DC PL1'9 '

16. Mostrar WK1 después UNPK WK1, R 17. Mostrar WK3 después UNPK WK3, R

7.7. La Instrucción MVZ

En el ejemplo anterior, incluso después de desempaquetar campo C, voy a tener un problema en la impresión, debido a el signo: C se muestra como 0180A porque X'C1 'es el equivalente EBCDIC de C'a. Así que tengo que de alguna manera quitar la señal. (En un capítulo posterior, cuando hablamos de la ED

(editar) la instrucción, le aprender a imprimir el signo. Por ahora, simplemente lo eliminará.)

Para eliminar el signo, hacemos uso de nuestro conocimiento de la zona y los bits numéricos. En primer lugar, vamos a volver a escribir este número en la forma vertical, ya vimos anteriormente:

```
F F F F C          0 1 8 0 1
```

Y considerar el siguiente campo:

```
ZEROES DC CL5'00000 '          F F F F F          0 0 0 0 0
```

Si me encuentro en ceros a campo de CI, por supuesto, obtener todos los ceros. Pero lo que si me movía sólo la zona de parte de ceros? Me gustaría conseguir el texto siguiente:

```
F F F F F <- cambiado          0 1 8 0 1 <- sin cambios
```

Tenga en cuenta que iba a cambiar el signo de C (positivo) a F (sin firma). Este tipo de movimiento es consigue utilizando el MVZ (Move Zone) la instrucción. Por ejemplo MVZ C, los ceros

Por supuesto, es sólo el último (a la derecha la mayoría) de bytes que tengo que cambiar, así que esto podría ser codificado como MVZ C 4 (1), = X'F0 '

Si por alguna razón usted cambia la longitud del campo C (de CL5), tendrá que acordarse de cambiar el desplazamiento explícito (4) en esta instrucción. Hay una manera muy inteligente a este código la instrucción para que usted no tiene que preocuparse por la longitud del campo C. Deje el ensamblador determinar que para usted en el tiempo de montaje con el operador de duración: MVZ C + L'C-1 (1), = X'F0 ' El ensamblador se utilizará la definición de C para determinar que L'C = 5, y por lo tanto L'C-1 = 5-1 = 4.

El resultado final, X'F0F1F8F0F1 ', se imprimirá como 01801. ¿Qué pasa con el cero a la izquierda: posición de vanguardia ceros suelen ser suprimida? La edición instrucción que veremos en el siguiente capítulo eliminar los ceros a la izquierda, así como nos permiten imprimir el signo adecuado.

Una instrucción similar, MVN, se moverán los bits numéricos. Por ejemplo, si hubiera utilizado MVN C, los ceros que obtendría C igual

```
F F F F C <- sin cambios          0 0 0 0 0 <- cambiado
```

intenta Dado P DC PL3'314 '

18. Mostrar la representación hexadecimal de P. (Recuerde que cuando un campo de relleno se define con un DC, el valor inicial es siempre firmado.)

19. Escriba las instrucciones necesarias para "mover" P a WK4 tal que WK4 es X'F0F3F1F4.

20. Escriba las instrucciones necesarias para "mover" P a WK2 tal que WK2 es X'F1F4.

7.8. Ejemplo de programación: La producción de la Crónica de Ventas

Hemos hablado de Bolsas de formato a los números decimales y miró a las siguientes instrucciones: PACK, UNPK, AP, ZAP, y MVZ. Ahora estamos listos para volver a nuestro problema de programación: a producir la Crónica de ventas de Industrias Dindón.

El diseño de registro para COGS.DAT se mostró en de la primera página de este capítulo. Vamos a utilizar la definición de registro siguiente:

```
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL41        1-41 Inventory record
IDESC    DS    CL10         1-10 Product description
ICALIF   DS    CL3         11-13 Units sold in Calif
IILL     DS    CL3         14-16 Units sold in Illinois
IUTAH    DS    CL3         17-19 Units sold in Utah
IWISC    DS    CL3         20-22 Units sold in Wisconsin
IBEGIN   DS    CL3         23-25 Beginning inventory
IPURCH   DS    CL3         26-28 Purchases throughout year
IQOH     DS    CL3         29-31 Actual quantity on hand
ICOST    DS    CL4         32-35 Cost (each) 99V99
ISELL    DS    CL4         36-39 Sell for (each) 99V99
ICRLF    DS    CL2         40-41 PC/370 only - CR/LF
```

El diseño de impresión de la Crónica de ventas es el siguiente:

```
1 2 3 4 5 6          123456789012345678901234567890123456789012345678901234567890
                                    Ventas Crónica
         Producto Calif malos Utah Wisconsin TOTAL
---------- ----- ----- ----- ----- -----

XXXXXXXXXX XXX XXX XXX XXX XXX
XXXXXXXXXX XXX XXX XXX XXX XXX
XXXXXXXXXX XXX XXX XXX XXX XXX
         XXX registros procesados. ...
```

así que vamos a utilizar la siguiente definición de registro de salida:

```
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62        1-62
```

```
ODESC    DS    CL10         1-10 Product description
         DS    CL5         11-15
OCALIF   DS    CL3         16-18 Units sold in Calif
         DS    CL6         19-24
OILL     DS    CL3         25-27 Units sold in Illinois
         DS    CL6         28-33
OUTAH    DS    CL3         34-36 Units sold in Utah
         DS    CL6         37-42
OWISC    DS    CL3         43-45 Units sold in Wisconsin
         DS    CL6         46-51
OTOTAL   DS    CL3         52-54 Units sold in all states
         DS    CL6         55-60
OCRLF    DS    CL2         61-62 PC/370 only - CR/LF
```

En este ejemplo, la entrada y salida para las ventas por el estado son del mismo tamaño, por lo que no hay necesidad de PACK a ellos para imprimirlos, la instrucción MVC funciona bien:

```
MVC OCALIF, ICALIF
  MVC Oill, IILL
  MVC OUTAH, IUTAH
  MVC OWISC, IWISC
```

Pero para llegar OTOTAL, el total de unidades vendidas en todos los estados, se tendrá que llevar cada una de las ventas del estado cifras, a continuación, agregue a un total (también con embalaje), después de que UNPK total en OTOTAL, y eliminar el signo. Por lo tanto, la definición de:

```
Unidades WCALIF DC PL2'0 'se venden en California
SERÁ Unidades DC PL2'0 "vendidos en Illinois          Unidades WUTAH DC PL2'0 "vendidos
```

El código para encontrar las ventas totales es:

```
PACK WCALIF,ICALIF     Each product's sales must
PACK WILL,IILL          be packed so they can be
PACK WUTAH,IUTAH         added to total for this
PACK WWISC,IWISC          product...
ZAP  WTOTAL,=P'0'      Initialize the total to zero
AP   WTOTAL,WCALIF      and start adding...
AP   WTOTAL,WILL
AP   WTOTAL,WUTAH
AP   WTOTAL,WWISC
UNPK OTOTAL,WTOTAL     Move total to output,
MVZ  OTOTAL+2(1),=X'F0' and remove the sign.
```

Mover total de la producción, MVZ OTOTAL 2 (1), = X'F0 'y eliminar el signo. En este programa en particular que no era necesario llevar cada uno de

los estados las cifras de ventas en un aparte campo. Podría haber usado un campo de trabajo único (como PK2) como sigue:

```
ZAP  WTOTAL,=P'0'       Initialize total to zero
PACK PK2,ICALIF         Pack...
AP   WTOTAL,PK2         ...then add Calif to total
PACK PK2,IILL           Pack...
AP   WTOTAL,PK2         ...then add Illinois to total
PACK PK2,IUTAH          Pack...
AP   WTOTAL,PK2         ...then add Utah to total
PACK PK2,IWISC          Pack...
AP   WTOTAL,PK2         ...then add Wisc to total
UNPK OTOTAL,WTOTAL      Move total to output,
MVZ  OTOTAL+2(1),=X'F0'  and remove the sign.
```

La decisión de utilizar un campo de pic nic dedicado para cada campo de entrada (como en el ejemplo anterior) o para Bolsas de usar un campo único y compartido por todos los campos de entrada (como en el último ejemplo) dependerá de lo que vas a hacer con los datos: si usted va a necesitar el número varias veces, el uso de un dedicado Bolsas de campo para cada campo de entrada le salvará de tener que empacar cada campo de más de una vez.

Con el fin de determinar el número de registros procesados, vamos a modificar la rutina como LEA siguiente:

```
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
```

156 ⟨*search..., read INVENTRY into IREC, EOFSW,IN++ SVREAD return* 156⟩≡ (157 164 207 211)

```
READ     EQU   *
         ST    R10,SVREAD
         GET   INVENTRY,IREC              Read a single teacher record
         AP    #IN,=P'1'                  Increment record count
         B     READX
ATEND    EQU   *
         MVI   EOFSW,C'Y'
READX    EQU   *
         L     R10,SVREAD
         BR    R10
```

donde

```
#IN DC PL2'0' Input record count
```

En esta cuenta se muestra sólo una vez, después de todos los registros han sido procesados, la lógica de hacerlo aparecen en la sección Wrapup. Vamos a volver a utilizar el área de registro de salida de la siguiente manera:

```
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                  after all records have been processed.       *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         MVC   OREC,BLANKS
         MVC   OCRLF,WCRLF          PC/370 only.
         BAL   R10,WRITE            Skip a line.
         MVC   OREC(22),=CL22'XXX records processed.'
         UNPK OREC(3),#IN           Count
         MVZ   OREC+2(1),=X'F0'     Remove sign
         BAL   R10,WRITE
         CLOSE INVENTRY
         CLOSE REPORT
         WTO   'COGS7A ... Sales recap on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
```

El programa completo, COGS7A.MLC, sigue.

157 ⟨*COGS7A.MLC* 157⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: COGS7A.MLC                                   *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce report for COGSWORTH INDUSTRIES      *
*                    showing sales by state.                    *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'COGS7A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'COGS7A ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
⟨search..., setup ascii INVENTRY and REPORT, HDGS, prime READ, SVSETUP return 206⟩
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU   *
```

```
         ST    R10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         PUT   REPORT,HD5
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record. *
*****************************************************************
PROCESS  EQU    *
         ST    R10,SVPROC
         BAL   R10,FORMAT
         BAL   R10,WRITE
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
⟨search..., read INVENTRY into IREC, EOFSW,IN++ SVREAD return 156⟩
*****************************************************************
*        FORMAT - Format a single detail line.                  *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORM
         MVC   OREC,BLANKS
         MVC   ODESC,IDESC
         MVC   OCALIF,ICALIF
         MVC   OILL,IILL
         MVC   OUTAH,IUTAH
         MVC   OWISC,IWISC
         PACK WCALIF,ICALIF              Each product's sales must
         PACK WILL,IILL                    be packed so they can be
         PACK WUTAH,IUTAH                    added to total for this
         PACK WWISC,IWISC                     product...
         ZAP   WTOTAL,=P'0'              Initialize the total to zero
         AP    WTOTAL,WCALIF               and start adding...
         AP    WTOTAL,WILL
         AP    WTOTAL,WUTAH
         AP    WTOTAL,WWISC
         UNPK OTOTAL,WTOTAL              Move total to output,
         MVZ   OTOTAL+2(1),=X'F0'         and remove the sign.
         MVC   OCRLF,WCRLF               PC/370 only.
         L     R10,SVFORM
```

```
         BR    R10
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
⟨search..., write REPORT from OREC, no line counting, SVWRITE return 106⟩
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         MVC   OREC,BLANKS
         MVC   OCRLF,WCRLF                PC/370 only.
         BAL   R10,WRITE                  Skip a line.
         MVC   OREC(22),=CL22'XXX records processed.'
         UNPK OREC(3),#IN                 Count
         MVZ   OREC+2(1),=X'F0'           Remove sign
         BAL   R10,WRITE
         CLOSE INVENTRY
         CLOSE REPORT
         WTO   'COGS7A ... Sales recap on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                   PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                    End of file? (Y/N)
BLANKS   DC    CL62' '
WCALIF   DC    PL2'0'                    Units sold in Calif
WILL     DC    PL2'0'                    Units sold in Illinois
```

```
WUTAH    DC    PL2'0'                    Units sold in Utah
WWISC    DC    PL2'0'                    Units sold in Wisconsin
WTOTAL   DC    PL2'0'                    Units sold in all states
#IN      DC    PL2'0'                    Input record count
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL41         1-41 Inventory record
IDESC    DS    CL10          1-10 Product description
ICALIF   DS    CL3          11-13 Units sold in Calif
IILL     DS    CL3          14-16 Units sold in Illinois
IUTAH    DS    CL3          17-19 Units sold in Utah
IWISC    DS    CL3          20-22 Units sold in Wisconsin
IBEGIN   DS    CL3          23-25 Beginning inventory
IPURCH   DS    CL3          26-28 Purchases throughout year
IQOH     DS    CL3          29-31 Actual quantity on hand
ICOST    DS    CL4          32-35 Cost (each) 99V99
ISELL    DS    CL4          36-39 Sell for (each) 99V99
ICRLF    DS    CL2          40-41 PC/370 only - CR/LF
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62         1-62
ODESC    DS    CL10          1-10 Product description
         DS    CL5          11-15
OCALIF   DS    CL3          16-18 Units sold in Calif
         DS    CL6          19-24
OILL     DS    CL3          25-27 Units sold in Illinois
         DS    CL6          28-33
OUTAH    DS    CL3          34-36 Units sold in Utah
         DS    CL6          37-42
OWISC    DS    CL3          43-45 Units sold in Wisconsin
         DS    CL6          46-51
OTOTAL   DS    CL3          52-54 Units sold in all states
         DS    CL6          55-60
OCRLF    DS    CL2          61-62 PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
HD1      DS    0CL62
         DC    CL40'                 COGSWORTH INDUSTRIES  '
         DC    CL20' '
         DC    XL2'0D25'
HD2      DS    0CL62
         DC    CL40'                      Sales Recap      '
         DC    CL20' '
```

```
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'Product        Calif     Ill       Utah '
         DC    CL20' Wisc      TOTAL'
         DC    XL2'0D25'
HD5      DS    0CL62
         DC    CL40'-----     ---    ---    ---  '
         DC    CL20' ---     ---'
         DC    XL2'0D25'
         END   BEGIN
```

7.9. Ejemplo de programación: La producción de las discrepancias Informe de Inventario

Nuestro próximo informe, las discrepancias de inventario, aparecerá como sigue:

```
 1 2 3 4 5 6          12345678901234567890123456789012345678901234567890123456789012345
```

Observe que la diferencia (resultado) es esperar-real, y esta diferencia se imprime sólo si otros que cero. Por lo tanto, a fin de completar este programa, se necesitan dos instrucciones más: SP (restar envasados) y CP (compárese con embalaje).

7.10. La Instrucción SP

El SP (restar envasados) la instrucción se utiliza para restar un número de pic nic de la otra. Para ejemplo, SP A, B resta de la A a B, con B sin cambios y la diferencia en el A.

Considere el siguiente ejemplo. Habida cuenta de estas definiciones de campo:

```
REAL DS CL6          PRESUPUESTO DS CL6          DIFERIR DS CL6 ... restar reales de dar
```

(Nótese el uso de PKN campos de trabajo tipo, la definición de lo que debería ser obvio por nuestro anterior discusión. PRESUPUESTO podría caber en un campo de cuatro bytes, pero ya estaba PK4 retención real, así que usé PK5 en su lugar.)

No debe haber cierta preocupación por el hecho de que simplemente quitar el signo de que son diferentes. Ciertamente, tener más de presupuesto no es el mismo que en el presupuesto! Nosotros nos ocuparemos de este problema en el siguiente capítulo.

intenta

Dado

```
A PL2'10 DC, DC PL3'5 B ', C PL1' DC-3', y D DC PL2'7.
```

Mostrar la representación hexadecimal de la A después de cada una de las siguientes. (Comience con nuevos datos cada vez.)

```
21. SP A, B,
22. SP A, C
23. SP D, A
```

7.11. La Instrucción CP

El PP (compárese con embalaje) instrucción funciona igual que el CLC (compárese con carácter lógico) la instrucción, salvo:

- ambos operandos deben ser válidos campos de pic nic, y
- La longitud máxima de cada operando es de 16.

El segmento siguiente programa ilustra el uso de SP y PP. Teniendo en cuenta:

```
Un PL3 DS          B DS PL3          C DS PL4 ...
```

estamos para restar el menor de A y B de C. El código de BAL necesario es el siguiente:

```
     CP  A,B
     BH  USEB
     SP  C,A
     B   DONE
USEB EQU *
     SP  C,B
DONE EQU *
```

intenta Dado A, B, C y D son válidos todos los campos de pic nic.

24. Escriba el código necesario para restar uno de A si A es mayor que B.
25. Escriba el código necesario para añadir la A a la B si B es igual a C.
26. Escriba el código necesario para mover el máximo de A, B y C a D.

7.12. Ejemplo de programación Revisited

Ahora estamos listos para volver a nuestro problema de programación: para producir las discrepancias de inventario Informe de Industrias Dindón. Vamos a utilizar la misma definición de registro de entrada que se utilizó en COGS7A.MLC. El diseño de impresión para el informe es el siguiente:

```
1 2 3 4 5 6          12345678901234567890123456789012345678901234567890123456789012345
```

así que vamos a utilizar la siguiente definición de registro de salida:

```
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL67        1-67
ODESC    DS    CL10         1-10 Product description
         DS    CL4         11-14
OBEGIN   DS    CL3         15-17 Beginning inventory
         DS    CL5         18-22
OPURCH   DS    CL3         23-25 Purchases
         DS    CL5         26-30
OSALES   DS    CL3         31-33 Units sold
         DS    CL6         34-39
OENDING  DS    CL3         40-42 Ending inventory (expected)
         DS    CL5         43-47
OQOH     DS    CL3         48-50 Ending inventory (actual)
         DS    CL6         51-56
ODIFF    DS    CL3         57-59 Difference
         DS    CL1         60-60
ORESULT  DS    CL5         61-65 'over' or 'short'
OCRLF    DS    CL2         66-67 PC/370 only - CR/LF
```

Nuestra lógica para el formato de las líneas de detalle es el siguiente:

1. Paso 1: Añadir California, Illinois, Utah, y las ventas de Wisconsin para obtener las ventas totales. Desde la entrada de los campos no están llenas, y puesto que requiere además ambos campos embalados, tendremos que utilizar algunos campos de trabajo.

```
PACK WCALIF,ICALIF     Each product's sales must
PACK WILL,IILL          be packed so they can be
PACK WUTAH,IUTAH         add to total for this
PACK WWISC,IWISC          product...
ZAP  WTOTAL,=P'0'      Initialize the total to zero
AP   WTOTAL,WCALIF      and start adding...
```

```
AP   WTOTAL,WILL
AP   WTOTAL,WUTAH
AP   WTOTAL,WWISC
UNPK OSALES,WTOTAL     Move total to output,
MVZ  OSALES+2(1),=X'F0' and remove the sign.
```

2. Paso 2: El inventario final esperado es igual al inventario inicial más las compras realizadas Desde entonces, menos las ventas totales. Esto también requerirá el uso de algunos campos de trabajo.

```
PACK WBEGIN,IBEGIN
PACK WPURCH,IPURCH      Expected ending inventory =
ZAP  WENDING,WBEGIN      Beginning
AP   WENDING,WPURCH       + Purchases
SP   WENDING,WTOTAL        - Sales
UNPK OENDING,WENDING
MVZ  OENDING+2(1),=X'F0'
```

3. Paso 3: Reste la cantidad real de la mano de el inventario final esperado. Muestra este diferencia, si no es igual a cero. Incremento de los contadores para el número de registros que indiquen extraedad o escasez.

```
        PACK WQOH,IQOH
        CP   WQOH,WENDING       Compare actual vs. expected
        BE   FORMATX            Don't show difference if zero
        BL   SHORT
        AP   #OVER,=P'1'        Count overages
        MVC  ORESULT,=CL5'over'
        B    DODIFF
SHORT   EQU  *
        AP   #SHORT,=P'1'       Count shortages
        MVC  ORESULT,=CL5'short'
DODIFF  EQU  *
        ZAP  WDIFF,WENDING      Difference = Expected - Actual
        SP   WDIFF,WQOH
        UNPK ODIFF,WDIFF
        MVZ  ODIFF+2(1),=X'F0'
FORMATX EQU  *
```

Cuenta Registro se muestran en la sección. Este código se incluye en el siguiente (completo)

Wrapup listado del programa:

164 ⟨*COGS7B.MLC* 164⟩≡

```
        PRINT NOGEN
*****************************************************************
```

```
*        FILENAME: COGS7B.MLC                                          *
*        AUTHOR : Bill Qualls                                          *
*        SYSTEM : PC/370 R4.2                                          *
*        REMARKS : Produce report for COGSWORTH INDUSTRIES             *
*                    showing inventory discrepancies.                  *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'COGS7B ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'COGS7B ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,              *
*                  before any records are processed.                   *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     INVENTRY+10,X'08'          PC/370 ONLY - Convert all
*                                          input from ASCII to EBCDIC
         OI     REPORT+10,X'08'            PC/370 ONLY - Convert all
*                                          output from EBCDIC to ASCII
         OPEN INVENTRY
         OPEN REPORT
         BAL    R10,HDGS
         BAL    R10,READ
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                        *
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
         PUT    REPORT,HD5
         L      R10,SVHDGS
         BR     R10
*****************************************************************
*        PROCESS - Those things which happen once per record. *
*****************************************************************
PROCESS EQU     *
         ST     R10,SVPROC
```

```
                BAL    R10,FORMAT
                BAL    R10,WRITE
                BAL    R10,READ
                L      R10,SVPROC
                BR     R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
⟨search..., read INVENTRY into IREC, EOFSW,IN++ SVREAD return 156⟩
          *****************************************************************
          *        FORMAT - Format a single detail line.                                  *
          *****************************************************************
          FORMAT   EQU    *
                   ST     R10,SVFORM
                   MVC    OREC,BLANKS
                   MVC    ODESC,IDESC              Description
                   MVC    OBEGIN,IBEGIN            Beginning inventory
                   MVC    OPURCH,IPURCH            Purchases
                   PACK WCALIF,ICALIF              Each product's sales must
                   PACK WILL,IILL                    be packed so they can be
                   PACK WUTAH,IUTAH                    added to total for this
                   PACK WWISC,IWISC                     product...
                   ZAP    WTOTAL,=P'0'             Initialize the total to zero
                   AP     WTOTAL,WCALIF              and start adding...
                   AP     WTOTAL,WILL
                   AP     WTOTAL,WUTAH
                   AP     WTOTAL,WWISC
                   UNPK OSALES,WTOTAL              Move total to output,
                   MVZ    OSALES+2(1),=X'F0' and remove the sign.
                   PACK WBEGIN,IBEGIN
                   PACK WPURCH,IPURCH              Expected ending inventory =
                   ZAP    WENDING,WBEGIN             Beginning
                   AP     WENDING,WPURCH               + Purchases
                   SP     WENDING,WTOTAL                - Sales
                   UNPK OENDING,WENDING
                   MVZ    OENDING+2(1),=X'F0'
                   MVC    OQOH,IQOH                Actual ending inventory
                   MVC    OCRLF,WCRLF              PC/370 only.
                   PACK WQOH,IQOH
                   CP     WQOH,WENDING             Compare actual vs. expected
                   BE     FORMATX                  Don't show difference if zero
                   BL     SHORT
                   AP     #OVER,=P'1'              Count overages
                   MVC    ORESULT,=CL5'over'
                   B      DODIFF
          SHORT    EQU    *
```

```
         AP    #SHORT,=P'1'              Count shortages
         MVC   ORESULT,=CL5'short'
DODIFF   EQU   *
         ZAP   WDIFF,WENDING             Difference = Expected - Actual
         SP    WDIFF,WQOH
         UNPK ODIFF,WDIFF
         MVZ   ODIFF+2(1),=X'F0'
FORMATX EQU    *
         L     R10,SVFORM
         BR    R10
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,OREC               Write report line
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         MVC   OREC,BLANKS
         MVC   OCRLF,WCRLF                PC/370 only.
         BAL   R10,WRITE                 Skip a line.
         MVC   OREC(22),=CL22'XXX records processed.'
         UNPK OREC(3),#IN                Count
         MVZ   OREC+2(1),=X'F0'          Remove sign
         BAL   R10,WRITE
         MVC   OREC(22),=CL22'XXX indicate shortage.'
         UNPK OREC(3),#SHORT             Count
         MVZ   OREC+2(1),=X'F0'          Remove sign
         BAL   R10,WRITE
         MVC   OREC(22),=CL22'XXX indicate overage. '
         UNPK OREC(3),#OVER              Count
         MVZ   OREC+2(1),=X'F0'          Remove sign
         BAL   R10,WRITE
         CLOSE INVENTRY
         CLOSE REPORT
         WTO   'COGS7B ... Discrepancies report on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
```

```
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
REPORT   DCB   LRECL=67,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
```

⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩

```
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                   End of file? (Y/N)
BLANKS   DC    CL67' '
WCALIF   DC    PL2'0'                   Units sold in Calif
WILL     DC    PL2'0'                   Units sold in Illinois
WUTAH    DC    PL2'0'                   Units sold in Utah
WWISC    DC    PL2'0'                   Units sold in Wisconsin
WTOTAL   DC    PL2'0'                   Units sold in all states
WBEGIN   DC    PL2'0'                   Beginning inventory
WPURCH   DC    PL2'0'                   Purchases
WENDING DC     PL2'0'                   Ending inventory (expected)
WQOH     DC    PL2'0'                   Ending inventory (actual)
WDIFF    DC    PL2'0'                   Difference
#IN      DC    PL2'0'                   Input record count
#OVER    DC    PL2'0'                   Records showing overage
#SHORT   DC    PL2'0'                   Records showing shortage
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL41         1-41 Inventory record
IDESC    DS    CL10          1-10 Product description
ICALIF   DS    CL3          11-13 Units sold in Calif
IILL     DS    CL3          14-16 Units sold in Illinois
IUTAH    DS    CL3          17-19 Units sold in Utah
IWISC    DS    CL3          20-22 Units sold in Wisconsin
IBEGIN   DS    CL3          23-25 Beginning inventory
IPURCH   DS    CL3          26-28 Purchases throughout year
IQOH     DS    CL3          29-31 Actual quantity on hand
ICOST    DS    CL4          32-35 Cost (each) 99V99
ISELL    DS    CL4          36-39 Sell for (each) 99V99
```

```
ICRLF    DS    CL2          40-41 PC/370 only - CR/LF
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL67         1-67
ODESC    DS    CL10          1-10 Product description
         DS    CL4          11-14
OBEGIN   DS    CL3          15-17 Beginning inventory
         DS    CL5          18-22
OPURCH   DS    CL3          23-25 Purchases
         DS    CL5          26-30
OSALES   DS    CL3          31-33 Units sold
         DS    CL6          34-39
OENDING DS     CL3          40-42 Ending inventory (expected)
         DS    CL5          43-47
OQOH     DS    CL3          48-50 Ending inventory (actual)
         DS    CL6          51-56
ODIFF    DS    CL3          57-59 Difference
         DS    CL1          60-60
ORESULT DS     CL5          61-65 'over' or 'short'
OCRLF    DS    CL2          66-67 PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
HD1      DS    0CL67
         DC    CL40'                    COGSWORTH INDUSTRIES'
         DC    CL25' '
         DC    XL2'0D25'
HD2      DS    0CL67
         DC    CL40'               Inventory Discrepancies R'
         DC    CL25'eport'
         DC    XL2'0D25'
HD3      DS    0CL67
         DC    CL65' '
         DC    XL2'0D25'
HD4      DS    0CL67
         DC    CL40'Product       Begin + Purch - Sales = Exp'
         DC    CL25'ect    Actual     Result '
         DC    XL2'0D25'
HD5      DS    0CL67
         DC    CL40'-----    ---   ---   ---   --'
         DC    CL25'--    ---   -----'
         DC    XL2'0D25'
         END   BEGIN
```

7.13. El SS formato de instrucción Revisited ...

El PACK, UNPK, AP, ZAP, SP, CP y las instrucciones son de todo tipo SS, o de almacenamiento a almacenamiento, instrucciones. Cuando se introdujo por primera vez el formato de instrucción de las SS, le dijo que no había más de un forma de instrucción de las SS. Todas estas instrucciones son de la segunda forma:

```
+--------------+-------------+--------------+              | 1 de halfword | 2 halfwo
```

La forma diferente es necesario porque estas operaciones tienen dos campos de longitud, es decir, la operandos pueden ser de diferentes longitudes. (Recuérdese que el MVC y las instrucciones de la CVX tiene una longitud de único operador y que la duración de la operación está determinada por la longitud del primer operando.)

Como con todas las instrucciones de ensamblador, el primero de los seis octetos es el código de operación. Estos op los códigos son los siguientes:

```
Op. Código de Instrucción                                    PACK X'F2 '
```

(MVZ MVN y también instrucciones de tipo SS, pero de la misma forma que el MVC y las instrucciones de la CVX. El código de operación para MVZ es X'D3. El código de operación para MVN es X'D1 '.)

Usted puede ver la mayoría de estos códigos op en el siguiente extracto de COGS7B.PRN:

```
0001C6 F212D472D498   0482   04A8 161         PACK WQOH,IQOH
0001CC F911D472D470   0482   0480 162         CP   WQOH,WENDING
0001D2 4780D1FE              020E 163         BE   FORMATX
0001D6 4740D1DA              01EA 164         BL   SHORT
0001DA FA10D478D2FC   0488   030C 165         AP   #OVER,=P'1'
0001E0 D204D4E1D2F5   04F1   0305 166         MVC  ORESULT,=CL5'over'
0001E6 47F0D1E6              01F6 167         B    DODIFF
0001EA              000001EA      168 SHORT   EQU  *
0001EA FA10D47AD2FC   048A   030C 169         AP   #SHORT,=P'1'
0001F0 D204D4E1D2F0   04F1   0300 170         MVC  ORESULT,=CL5'short'
0001F6              000001F6      171 DODIFF  EQU  *
0001F6 F811D474D470   0484   0480 172         ZAP  WDIFF,WENDING
0001FC FB11D474D472   0484   0482 173         SP   WDIFF,WQOH
000202 F321D4DDD474   04ED   0484 174         UNPK ODIFF,WDIFF
000208 D300D4DFD2FA   04EF   030A 175         MVZ  OD
00020E              0000020E      176 FORMATX EQU *
```

El segundo byte de la instrucción es la longitud. Cada longitud ocupa un half-byte, o cuatro bits. Recordemos que cuatro bits puede variar en el valor de 0

(todos los ceros) a 15 (todos los). Pero así como en nuestro la discusión de MVC y CLC, un campo de longitud 0, no tiene ningún sentido. Así que las longitudes son en realidad de la longitud de cada operando menos uno. De esta manera, los valores de 0 a 15 indican longitudes de operación de (suma, resta, compara, etc), de 1 a 16. Por ello, estas operaciones son decimal empaquetado limitada a los campos de longitud de 1 a 16.

Por ejemplo, en la línea 161 que PACK IQOH en WQOH. WQOH se definió como PL2, por lo que la longitud de la primera ser 2-1 = 1. IQOH se definió como CL3, por lo que la longitud del segundo será 3-1 = 2. Es por eso que el código de operación de X'F2 'es seguido por X'12'.

Bytes El halfword segundo (tercero y cuarto) indica que la base y el desplazamiento de la primera octetos operando, y la media palabra tercero (quinto y sexto) indica que la base y el desplazamiento de la segundo operando, al igual que con MVC y la CVX. Nuestro análisis anterior de las columnas ADR1 y ADR2 se aplica también.

intenta 27. Mostrar el código de objeto si la línea 161 supra se cambiaron a UNPK IQOH, WQOH

28. Mostrar el código de objeto si la línea 169 supra se han cambiado a AP # CORTO, # que la

29. Mostrar el código de objeto si la línea 172 supra se han cambiado a AP wdiff, Wentling +1 (1)

30. Mostrar el código de objeto si la línea 174 supra se cambiaron a UNPK ODIFF (2), comienzan a cursar

31. Mostrar el código de objeto si la línea 175 supra se cambiaron a MVN ODIFF 2 (1), = X'F0 '

32. Mostrar el código de objeto si la línea 175 supra se cambiaron a MVZ ODIFF +1 (1), = X'F0 '

7.14. Datos de errores de excepción

Ahora que usted está trabajando con Bolsas de campos de decimales, tienen una probabilidad mucho mayor de tener los errores de tiempo de ejecución. Estos se refieren a veces como abends para finales anormal (o anormal de terminación.) Con mucho, el error de este tipo más común es el error de excepción de los datos. Esto ocurre cada vez que se intenta realizar una operación aritmética decimal envasados con operandos que son no números válidos de pic nic. Por ejemplo, si usted se olvida de empacar un número antes de añadirlo a una acumulador, su programa se Abend. En el mundo mainframe de IBM esto produce un sistema de 0C7 código de error, o S0C7 (leer como "calcetín siete"). PC/370 también producir un error, aunque no es llamado S0C7.

Recordemos que la AP, ZAP, SP, y CP se consideran las operaciones aritméticas: PACK y UNPK no lo son. El el primer operando de ZAP puede ser cualquier cosa, pero el segundo operando debe ser un decimal empaquetado

válido número. PACK UNPK y nunca se produce un S0C7, pero se olvidan de PACK o UNPK puede ciertamente!

PC/370tiene una prueba / centro depurador, pero es muy primitivo. Sin embargo, si usted sabe lo que buscar que puede ser muy útil. Considere el siguiente programa, S0C7.MLC:

```
         START 0
BEGIN    BEGIN
* --------------------------
*   S0C7.MLC BY BILL QUALLS
*   USING PC/370 V4.2
*   FORCE DECIMAL EXCEPTION
* --------------------------
         AP     SUM,ONE
         AP     SUM,TWO
         RETURN
SUM      DC     PL2'0'
ONE      DC     PL1'1'
TWO      DS     PL1'2' (Error)
         END
```

Este programa contiene un error deliberado: la definición de dos debería ser un DC en lugar de una DS. Cuando intento agregar dos a SUM (el segundo AP), voy a tener una Abend. Si nos reunimos, enlaces, y ejecutar este programa se obtiene la siguiente:

```
A:\MIN>s0c7
TRACE EP A=08E4 ID=BUG   370 A=000276 OP=58DD0004
 ***********************************************
 *    PC/370 System Release 4.2    01/07/88    *
 *    Copyright (C) 1988 Donald S. Higgins     *
 *                                             *
 * You are encouraged to copy and share this   *
 * package with other users on the condition   *
 * the package is not distributed in modified *
 * form, and that no fee is charged. If you    *
 * find PC/370 useful, send 45 dollars to the *
 * address below to become registered user and *
 * support continued shareware development.    *
 * Registered users will receive notices of    *
 * future PC/370 releases.                     *
 *                                             *
 *      Don Higgins                            *
 *      6365 - 32 Avenue, North                *
 *      St. Petersburg, Florida 33710          *  ************************************
TYPE H FOR HELP
Si tipo H (pero no pulse Enter), obtenemos la pantalla de ayuda siguientes:
```

```
A  SELECT ADDRESS STOP OPTIONS
C  CONTINUE TO NEXT TRACE ID
D  DUMP 370/386 MEMORY AT SELECTED ADDRESS
E  SET EBCDIC OR ASCII DUMP FORMAT
F  SET FIND TRACE ID
I  DISPLAY 370 INSTRUCTION COUNTER WORD
J  JUMP TO NEW 370/386 INSTRUCTION ADDRESS
K  SET KILL TRACE MODE OR RESTORE TRACES
L  SET LIMIT FOR Q AND T
M  MODIFY 370/386 MEMORY
N  LIST TRACE TABLE
P PRINTER ON/OFF
Q SET QUIET MODE
R LIST 370/386 REGISTERS
S SAVE/UNSAVE CURRENT TRACE ID FROM KILL
T SET TRACE MODE
W LIST 370/386 FREE STORAGE
X ASSIST LOG ON/OFF
Y MODIFY 370/386 REGISTER
Z SWITCH 370/386 MODE (AFFECTS D,E,J,O,M,R,W,Y)
<CR>=REPEAT DUMP, <BS>=DUMP BACK, <SP>=DUMP FORWARD
<ESC>=EXIT TO MS-DOS
```

(Pulsamos Esc para volver al DOS.)

nos dice que el programa ha terminado, y A = 000276 nos dice dónde. Pero la dirección de ID = ERROR (000276) no significa nada para nosotros sin la inclusión de la Asamblea: tenemos que tener el. PRN inclusión en el orden de de ser capaz de hacer cualquier depuración.

```
LOC                ADR1 ADR2 LINE LABEL    OP       OPERANDS
000000                             1          START    0
000000                             2 *++++    BEGIN
000000                             3 BEGIN    CSECT
000000                             4          USING    *,15
000000 47F0F058            0058    5          B        KZHQX001
000004 0B                          6          DC       AL1(11)
000005 C2C5C7C9D5404040            7          DC       CL11'BEGIN    '
000010 0000000000000000            8 HZQKX001 DC       18F'0'
000058 90ECD00C            000C    9 KZHQX001 STM      14,12,12(13)
00005C 50D0F014            0014   10          ST       13,HZQKX001+4
000060 18ED                        11          LR       14,13
000062 41D0F010            0010   12          LA       13,HZQKX001
000066 50D0E008            0008   13          ST       13,8(0,14)
00006A                             14          DROP     15
00006A                             15          USING    HZQKX001,13
```

```
00006A                              16 * ---------------------------
00006A                              17 *   S0C7.MLC BY BILL QUALLS
00006A                              18 *   USING PC/370 V4.2
00006A                              19 *   FORCE DECIMAL EXCEPTION
00006A                              20 * ---------------------------
00006A FA10D070D072   0080 0082    21          AP       SUM,ONE
000070 FA10D070D073   0080 0083    22          AP       SUM,TWO
000076                              23 ******** RETURN
000076 58DD0004            0004    24          L        13,4(13)
00007A 98ECD00C            000C    25          LM       14,12,12(13)
00007E 07FE                        26          BR       14
000080 000C                        27 SUM      DC       PL2'0'
000082 1C                          28 ONE      DC       PL1'1'
000083                             29 TWO      DS       PL1'2' (Error)
000000                             30          END
```

La dirección se refiere a la lucha contra la ubicación (LOC) en el extremo izquierdo. Pero nótese que no hay ninguna 000276! Restamos X'200 'de la dirección (siempre). Esto da una dirección de 000076, que corresponde a la línea 23. Pero espere! Cuando PC/370 se encuentra con un error de excepción de los datos, le da la dirección de la siguiente instrucción que se ha ejecutado el programa no había abended! Así que la instrucción real en el error está en la posición 000070, línea 22, la AP que dijimos haría que la Abend! Esa instrucción se nos hacen mirar a los valores de SUM y dos. SUM está bien (o no tendría abended con la instrucción anterior). Así que mirar y ver que DOS debería haber sido más que DC DS.

No es fácil. Y es aún peor cuando se trata de archivo de entrada. Lo que se necesita es de alguna manera a buscar en cualquier campo, mientras que el programa se está ejecutando. Para ello, debe ejecutar el programa en la prueba modo. Para ejecutar un programa en modo de prueba, el nombre del programa que se consignan, seguido de un blanco y un mayúsculas T. Por ejemplo,

```
A:\MIN>s0c7
TRACE EP A=08E4 ID=BUG   370 A=000276 OP=58DD0004
 ***********************************************
 *    PC/370 System Release 4.2    01/07/88    *
 *    Copyright (C) 1988 Donald S. Higgins     *
 *                                             *
 * You are encouraged to copy and share this   *
 * package with other users on the condition   *
 * the package is not distributed in modified  *
 * form, and that no fee is charged. If you    *
 * find PC/370 useful, send 45 dollars to the  *
 * address below to become registered user and *
 * support continued shareware development.    *
 * Registered users will receive notices of    *
```

```
 * future PC/370 releases.                    *
 *                                            *
 *      Don Higgins                           *
 *      6365 - 32 Avenue, North               *
 *      St. Petersburg, Florida 33710         *
 ********************************************
TYPE H FOR HELP
```

Queremos detener el programa antes de ofender a la instrucción se ejecuta, por lo que en el indicador (+) para que escriba una dirección. La instrucción está en la posición 000070 por lo que añadimos X'200 "dar 000270. Por lo tanto, entramos en 270 cuando se le pida la dirección. Cuando se le preguntó por el tipo de dirección, el tipo de A otra vez:

```
TYPE H FOR HELP
+a
ADDR STOP ON
A=270
    000270 FA10D070 D07358DD 000498EC D00C07FE ..........q.....
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
```

Ahora t tipo de rastro. Cada instrucción está en la lista antes de su ejecución.

```
+t
TRACE  SET
TRACE  EP A=1433 ID=BC    370   A=000200      OP=47F0F058
TRACE  EP A=1F9B ID=STM   370   A=000258      OP=90ECD00C
TRACE  EP A=17D1 ID=ST    370   A=00025C      OP=50D0F014
TRACE  EP A=0CAD ID=LR    370   A=000260      OP=18ED
TRACE EP A=1649 ID=LA   370 A=000262 OP=41D0F010
TRACE EP A=17D1 ID=ST   370 A=000266 OP=50D0E008
TRACE EP A=2093 ID=AP   370 A=00026A OP=FA10D070D072
ADDR STOP
    000270 FA10D070 D07358DD 000498EC D00C07FE ..........q.....
TRACE EP A=2093 ID=AP   370 A=000270 OP=FA10D070D073
+
```

Podemos ver que la siguiente instrucción a ejecutar es FA10D070D073. (Usted puede ver el código de este mismo en la línea 22 de la. lista PRN). De nuestro análisis de los formatos de instrucción, sabemos que este es un punto de acceso, y esto es confirmado por el mensaje de ID = AP. También podemos decir por el X'10 'tras la X'FA' que vamos a añadir un campo de un octeto de un campo de dos bytes. Recuerde: la instrucción no ha sido ejecutado todavía. Ahora puede mostrar los datos en cuestión antes de permitir que el programa para continuar. Para hacerlo, escriba d (por pantalla). Los datos en cuestión comienza en la posición 000.080, por lo que añadimos X'200 "dar 000280: que tipo 280 como la dirección cuando se le pida:

```
+d
A=280
    000280 001C1C00 00000000 00000000 0007C6F8 ............F8
    000290 00000000 00000000 00000000 00000000 ................
+
```

Podemos ver que suma, que se definió como PL2'0 ', ahora tiene un valor de 1 (X'001C'), por lo que sabemos que la AP trabajó por primera vez. Pero también podemos ver que dos tiene un valor de X'00 ', que no es válido número decimal de pic nic. Nosotros hemos encontrado el error. Queremos cambiar el. Código CTM, ensamblar, enlace, y ejecutar.

Con la práctica, s PC/370 'facilidad de prueba se convertirá en una herramienta de valor incalculable.

7.15. Ejercicios

1. Verdadero o falso. (Para cada una de las siguientes suponen una CL3'71D DC, DC CL3 B ', C DC PL2'-2 ', D DC PL2'3', DC CL2' y E-2 '. Comience con nuevos datos para cada pregunta.)

   a) La representación hexadecimal de A es X'F7F1C4. F T

   b) Los bits numéricos de A son 0111, 0001 y 0010. F T

   c) PACK B, B = A hace X'00071D. F T

   d) UNPK B, C, B = X'00002D hace. F T

   e) MVZ A 2 (1), = X'F0 'hace que A = X'F7F1FD'. F T

   f) AP C, D = D hace X'001C '. F T

   g) La representación hexadecimal de E es X'002D. F T

   h) SP D, C hace que D = X'005C '. F T

   i) SP D, D = D hace X'000C '. F T

   j) SP E, E = E hace X'000C '. F T

   k) CP C, D en los resultados de un código de condición de 4 (bajo). F T

   l) ZAP, B, C se producirá un Abend. F T

   m) ZAP B, E causará un Abend. F T

2. Determinar los resultados de las operaciones siguientes:

```
PACK   B,A                  A=     F1      F2       F3
      a.                                          B=              PACK   D,C
                                        C=     F1      F2       F3       C4
       b.
```

```
                                        D=              PACK   F,E
              E=     D1
        c.                              F=              PACK   H,G
             G=     F1      F2         F3       C4
        d.                              H=
```

3. Determinar los resultados de las operaciones siguientes:

```
UNPK   J,I                  I=     12      34       5F        a.
            UNPK   L,K                  K=     00      1C        b.
               M=     1D        c.                                      N=
            UNPK   P,O                  O=     12      3C        d.
```

4. Determinar los resultados de las operaciones siguientes:

```
AP     B,A                  A=     01      23       4F          a.
```

5. Determinar los resultados de las operaciones siguientes:

```
SP     B,A                  A=     01      23       4F          a.
```

6. Escriba el código de BAL necesarios para determinar donde TOTAL BRUTO TOTAL = + ImpuestoVenta. Utilice las definiciones de campo siguiente. Definir los campos de trabajo necesario. Total debe ser imprimibles, es decir, sin firmar.

```
GROSS    DS CL5
SALESTAX DS CL4
TOTAL    DS CL6
```

7. Escriba el código de BAL necesarias para determinar NET donde NET = Ventas - Devoluciones. Utilice las definiciones de campo siguiente. Definir los campos de trabajo necesario. NET debe imprimir, es decir, sin firmar.

```
SALES      DS      CL5
RETURNS    DS      CL4
NET        DS      CL5
```

8. Columnas 8-14 de una tarjeta de la imagen de archivo contienen saldo de la cuenta del cliente. Mostrar la y secciones de Wrapup de un programa que mostrará el número de clientes y de PROCESO del total (suma) de los saldos. Mostrar todas las definiciones de campo.

9. 9. Teniendo en cuenta F223D110D113 generados en LOC = 0000DA. PACK A, B,

a) ¿Cuánto dura un? B?

b) ¿Cuál es la LOC de la siguiente instrucción?

c) Mostrar el código objeto si esta línea se han cambiado a PACK B, A

d) Mostrar el código objeto si esta línea se ha cambiado a UNPK B 2 (2), A

10. Teniendo en cuenta generados en FA31D220D21E LOC = 0000C8. AP C, D

a) ¿Cuánto tiempo es C? D?

b) ¿Cuál es la LOC de la siguiente instrucción?

c) Mostrar el código objeto si esta línea se ha cambiado a ZAP D, C

d) Mostrar el código objeto si esta línea se cambió a SP C 1 (2), D +1 (1)

11. Escribir un programa que muestra un recuento del número de cursos ofrecidos en el semestre de W93. Utilice el archivo de la base de datos OFERTA Small Town Community College. Su salida debe ser por la OMC sólo: no hay ningún archivo de salida. Su mensaje debe aparecer como sigue:

```
There were XXX courses offered in semester W93.
```

12. Modificar el programa COGS7A.MLC para incluir totales por estado, es decir, su resultado debería aparecer como sigue:

```
1          2              3              4              5         6
                                      Sales Recap
Product          Calif         Ill           Utah          Wisc         TOTAL
----------       -----        -----         -----         -----         -----
GIZMOS            020          030           020           020           090
WIDGETS           015          010           010           002           037
JUNQUE            025          015           015           018           073
----------       -----        -----         -----         -----         -----
TOTAL             060          055           045           040           200
003 records processed.
```

13. Escribir un programa que producirá conteos por sexo y estado civil de los registros en el archivo. Su resultado debería aparecer como sigue:

```
      STUDENT
         1         2         3         4
12345678901234567890123456789012345678901234567890
STUDENT STATISTICS
```

```
Status          Male        Female       Total
--------       ------       ------       -----
Single           XXX          XXX          XXX
Married          XXX          XXX          XXX
--------       ------       ------       -----
Total            XXX          XXX          XXX
```

Note: A table of this type is called a "cross tabulation", or "cross tab".

14. 14.

a) (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Escribir un programa en el que figurarán sólo los elementos de la Herramienta de archivo que debe ser condenada. Se trata de artículos donde la suma de la cantidad en la mano y la cantidad de orden es menor o igual a la cantidad mínima. El informe debe aparecer como sigue:

```
1              2               3                4                5   6
           12345678901234567890123456789012345678901234567890123456789012345678901234567890
                                    SMALL TOWN HARDWARE STORE
TID            Description                    QOH + QOO = Sum                 Min
---      ----------------------               ---        ---       ---       ---
XXX      XXXXXXXXXXXXXXXXXXXXX               XXX        XXX       XXX       XXX
XXX      XXXXXXXXXXXXXXXXXXXXX               XXX        XXX       XXX       XXX
XXX      XXXXXXXXXXXXXXXXXXXXX               XXX        XXX       XXX       XXX
```

b) Modificar el programa en el inciso (a) de modo que, además de crear el informe, una nueva herramienta de archivo se crea con la cantidad en el campo para actualización de aquellos artículos que están ordenados. La nueva cantidad en el orden debe ser igual a la cantidad de la orden más antigua en la cantidad de orden económico. Utilice DDNAME = 'NEWTOOL.DAT' en el DCB para este nuevo archivo. Todas las herramientas deben ser escritos para este nuevo archivo, incluso aquellos para los que no había orden nuevo puesto. Comando USE DOS 'para ver el archivo.

15. Ejercicios 15.

a) a. Llene los espacios en blanco (líneas 15-19 y 26-29):

```
STUFF7A                                                                  PAGE
    PC/370 CROSS ASSEMBLER OPTIONS=LXACE
        LOC                    ADR1       ADR2 LINE       LABEL          OP
    000000                                             1   *******        BEGIN
    000000                                             2   STUFF7A        CSECT
    000000                                             3                  USING
```

```
000000 47F0F058                      0058        4                  B
000004 0B                                        5                  DC
000005 E2E3E4C6C6F7C140                          6                  DC
000010 0000000000000000                          7   HZQKX001 DC
000058 90ECD00C                      000C        8   KZHQX001 STM
00005C 50D0F014                      0014        9                  ST
000060 18ED                                     10                  LR
000062 41D0F010                      0010       11                  LA
000066 50D0E008                      0008       12                  ST
00006A                                          13                  DROP
00006A                                          14                  USING
00006A ____________          0097    0092       15                  PACK
000070 ____________          0097    0095       16                  AP
000076 ____D080____          0090    0097       17                  CP
00007C ________                      0086       18                  BE
000080 ____________          0095    0097       19                  SP
000086                   00000086                20   S              EQU
000086                                          21   *******  RETURN
000086 58DD0004                      0004       22                  L
00008A 98ECD00C                      000C       23                  LM
00008E 07FE                                     24                  BR
000090                                          25                  LTORG
000090 ____                                     26   M              DC
000092 ______                                   27   N              DC
000095 ____                                     28   Q              DC
000097 ________                                 29   R              DC
0000A0                                          30                  END
```

b) b. Usted es el equipo ... ¿cuáles son los valores de M, N, Q, R y después de este programa se ejecuta?

```
M=_________        N=_________        Q=_________        R=_________
```

c) c. Número y montamos este programa. Utilice PC/370 's laboratorio para detener el programa en S, y verificar sus respuestas a la parte (b). (Nota: Las líneas 1-14 del programa son la expansión de la macro de empezar, y las líneas de 21-24 son la expansión de la macro RETORNO: código de las declaraciones de macro como usted u

16. Ejercicios 16.

a) Llene los espacios en blanco (líneas 15-19 y 24-27):

```
STUFF7B                                                          PAGE
    PC/370 CROSS ASSEMBLER OPTIONS=LXACE
       LOC                   ADR1       ADR2 LINE       LABEL          OP
    000000                                          1   *******        BEGIN
```

```
000000                                                  2   STUFF7B         CSECT
000000                                                  3                   USING
000000 47F0F058                          0058           4                   B
000004 0B                                               5                   DC
       AL1(11)
000005 E2E3E4C6C6F7C240                                 6                   DC
000010 0000000000000000                                 7   HZQKX001 DC
000058 90ECD00C                          000C           8   KZHQX001 STM
00005C 50D0F014                          0014           9                   ST
000060 18ED                                            10                   LR
000062 41D0F010                          0010          11                   LA
000066 50D0E008                          0008          12                   ST
00006A                                                 13                   DROP
00006A                                                 14                   USING
00006A F823D08BD091          009B        00A1          15                   ------
000070 F322D08ED08B          009E        009B          16                   ------
000076 D300D090D088          00A0        0098          17                   ------
00007C D201D089D08F          0099        009F          18                   ------
000082 FB32D091D08B          00A1        009B          19                   ------
000088                                                 20   *+++++++ RETURN
000088 58DD0004                          0004          21                   L
00008C 98ECD00C                          000C          22                   LM
000090 07FE                                            23                   BR
000098                                                 24                   LTORG
000098 F0                                              24
000099                                                 25   W               ------
00009B F5F5F5                                          26   X               ------
00009E C1C2C3                                          27   Y               ------
0000A1 0001234C                                        28   Z               ------
0000A8                                                 29                   END
```

b) b. Usted es el equipo ... ¿qué son los valores de W, X, Y, Z, después de este programa se ejecuta?

```
W=_________        X=_________              Y=_________
                Z=_________
```

c) Número y montamos este programa. Utilice PC/370 's laboratorio para detener el programa antes de la vuelta, y verificar sus respuestas a la parte (b). (Nota: Las líneas 1-14 del programa son la expansión de la macro de empezar, y las líneas 20-23, la expansión de la macro RETORN: Código de esas declaraciones macro como lo harías normalmente.)

Capítulo 8

Lógica de Salto de página

Objetivos Al término de este capítulo usted será capaz de:

- Describir la lógica del código y la página de descanso, por ejemplo, ...
- Producir un avance de la impresora a la fuerza para avanzar hasta la parte superior de una página.

Introducción En el capítulo dos hemos visto cómo producir las partidas para nuestros informes. Dijimos en ese momento que nos sería posponer la discusión de los recuentos de páginas, porque cuenta como requiere el uso de la aritmética, que aún no había cubierto. Hasta ahora, todos nuestros informes se han limitado a una sola página de de salida, pero de haber discutido Bolsas de aritmética decimal, ahora estamos listos para producir "verdadera" partidas informe. En este capítulo analizaremos la lógica de salto de página, incluyendo la manera de producir las partidas con recuentos de página y cómo forzar a la impresora para avanzar hasta la parte superior de una nueva página.

* * * * * * * * * * * * * * * * * * * *

En este capítulo vamos a modificar TEACH2C.MLC del capítulo dos. Ese programa produjo una lista de todos los profesores de la pequeña ciudad universitaria de la Comunidad. Esta lista contenía el informe y los encabezados de columna, pero no incluye la lógica de salto de página. Vamos a modificar este programa para incluir las partidas con la página números. A fin de demostrar la lógica de salto de página mientras trabaja con un conjunto de datos de la de tamaño manejable, vamos a suponer que un máximo de tres líneas de detalle se van a imprimir por página. Nuestra nueva salida aparecerá como sigue:

```
           LIST OF TEACHERS     Page 001
ID#      Name          Degr Ten Phone
--- ---------------    ---- --- -----
732 BENSON, E.T.       PHD   N  5156
218 HINCKLEY, G.B.     MBA   N  5509
854 KIMBALL, S.W.      PHD   Y  5594
```

```
         LIST OF TEACHERS     Page 002
ID#      Name          Degr Ten Phone
--- ---------------   ---- --- -----
626 YOUNG, B.          MBA   Y  5664
574 SMITH, J.          MS    Y  5320
```

Hay cinco pasos para producir saltos de página: estos pasos se muestran en el diagrama de flujo en el siguiente página. Estos pasos requieren que agregar varios campos a nuestro programa:

```
PGS    DC PL2'0' Nbr of pages printed.
LNS    DC PL2'3' Lines printed on this page.
MAXLNS DC PL2'3' Max nbr lines per page.
*               My line counts exclude hdgs.
```

1. Antes de imprimir una línea, compruebe la línea de cuenta para ver si es mayor o igual a el número máximo de líneas permitido por página. Si es así, entonces ...
2. Añadir una al número de páginas,
3. Mueva el número de páginas a uno de los línea de rumbro
4. Imprimir las rubros
5. Reiniciar el conteo de líneas a cero.

Tenga en cuenta que el número de páginas (PGS) se inicializa a cero. El número máximo de líneas por página (MAXLNS) se inicializa con el máximo deseado. Dado que se trata de pequeños conjuntos de datos, y para fines de demostración, vamos a utilizar un máximo de tres. En el mundo real que probablemente uso 50-55. Por último, el recuento de línea (LNS) se inicializa con el valor máximo también. La razón de de hacerlo no puede ser evidente: lo hacemos de modo que la primera vez que comprobar el número de línea, el programa que la página actual está lleno, y por lo tanto, un salto de página.

Mi preferencia es contar sólo las líneas de detalle: las partidas no están incluidas en LNS y MAXLNS. Digo, "Los títulos son libres". Usted (o su empresa) puede preferir no hacerlo.

8.1. La rutina CHKLNS

En nuestro programa actual pondré el cheque conteo de líneas en la sección de proceso, después de FORMATO y antes de escribir. Voy a utilizar una rutina independiente con el fin de reducir al mínimo el número de sucursales (y etiquetas) dentro del proceso:

```
PROCESS EQU *
        ST  R10,SVPROC
        BAL R10,FORMAT
        BAL R10,CHKLNS
        BAL R10,WRITE
        BAL R10,READ
        L   R10,SVPROC
        BR  R10
```

donde CHKLNS se define como ...

```
*****************************************************************
*        CHKLNS - Check lines printed. Full page?               *
*****************************************************************
```

184 ⟨*search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS* 184⟩≡ (187 223 232 241 249 266 289)

```
CHKLNS   EQU   *
         ST    R10,SVCHKLNS
         CP    LNS,MAXLNS
         BL    CHKLNSX
         BAL   R10,HDGS
CHKLNSX  EQU    *
         L     R10,SVCHKLNS
         BR    R10
```

Agregar CHKLNS requiere la definición de un fullword para guardar la dirección de retorno:

```
SVCHKLNS DC F'0' CHKLNS
```

8.2. Incluyendo el número de página en las líneas de definición

A continuación, cambiar la definición de la línea de partida en primer lugar, HD1, para incluir el número de página:

```
      DS 0CL62
      DC CL40'     LIST OF TEACHERS Page '
HDPGS DC CL3' '
      DC CL17' '
      DC XL2'0D25'
```

No hay cambios necesarios para las líneas de partida.

8.2.1. La rutina HDGS

Por último, los pasos 2 a 5 se llevan a cabo dentro de la rutina HDGS:

```
HDGS EQU  *
     ST   R10,SVHDGS
     AP   PGS,=P'1'          Add 1 to page count
     UNPK HDPGS,PGS          Move page count to headings
     MVZ  HDPGS+L'HDPGS-1(1),=X'F0'    Remove sign
     PUT  REPORT,FORMFEED    PC/370 ONLY
     PUT  REPORT,HD1
     PUT  REPORT,HD2
     PUT  REPORT,HD3
     PUT  REPORT,HD4
     ZAP  LNS,=P'0'          Reset line count to zero
     L    R10,SVHDGS
     BR   R10
```

8.2.2. La Línea FORMFEED

Tenga en cuenta la adición de una declaración de aplicar la nueva: INFORME DE VENTA, FORMFEED. Es necesario indicar de alguna forma de que la impresora debe avanzar a la parte superior de la página siguiente. Una manera fácil de hacerlo es definir un línea de partida nueva que es:

- la misma longitud que las líneas de partida (62), y la
- consiste en todos los espacios en blanco,
- excepto el primer carácter, que es un carácter de avance.

Esto se hace de la siguiente manera:

```
FORMFEED DS 0CL62   PC/370 only
         DC X'0C'   EBCDIC formfeed
         DC CL59' '
         DC X'0D25' EBCDIC CR/LF
```

Un X'0C "es un salto de línea EBCDIC. (Por coincidencia, también es un salto de línea con ASCII). Mayoría de las impresoras de PC avanzarán a la parte superior de la página cuando reciben este personaje. Nota: esto no es como uno podría una fuerza de saltar al inicio de la página en el mundo mainframe: que se analiza en el apéndice.

Por supuesto, el recuento de línea debe ser incrementado después de escribir cada uno:

185 ⟨*search..., write REPORT from OREC, LNS++, SVWRITE return* 185⟩≡ (187 223 232 241)

```
WRITE    EQU   *
         ST    R10,SVWRITE
```

```
PUT   REPORT,OREC              Write report line
AP    LNS,=P'1'
L     R10,SVWRITE
BR    R10
```

8.2.3. Ejemplo de programa de solución

El programa completo, TEACH8A.MLC, sigue.

187 ⟨*TEACH8A.MLC* 187⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH8A.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : This is a revision of TEACH2C.MLC.           *
*                    Produce list of teachers, with headings.   *
*                    Introduce page break logic.                *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'TEACH8A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'TEACH8A ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,                    *
*                 before any records are processed.                          *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     TEACHERS+10,X'08' PC/370 ONLY - Convert all
*                                        input from ASCII to EBCDIC
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                        output from EBCDIC to ASCII
         OPEN TEACHERS
         OPEN REPORT
*                                          Note: Removed BAL R10,HDGS
         BAL    R10,READ
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         AP     PGS,=P'1'                 Add 1 to page count
         UNPK HDPGS,PGS                   Move page count to headings
         MVZ    HDPGS+L'HDPGS-1(1),=X'F0' Remove sign
         PUT    REPORT,FORMFEED           PC/370 ONLY
         PUT    REPORT,HD1
```

```
                PUT     REPORT,HD2
                PUT     REPORT,HD3
                PUT     REPORT,HD4
                ZAP     LNS,=P'0'                Reset line count to zero
                L       R10,SVHDGS
                BR      R10
        ************************************************************
        *       PROCESS - Those things which happen once per record.  *
        ************************************************************
        PROCESS EQU     *
                ST      R10,SVPROC
                BAL     R10,FORMAT
                BAL     R10,CHKLNS
                BAL     R10,WRITE
                BAL     R10,READ
                L       R10,SVPROC
                BR      R10
***************************************************************
*       READ - Read a record.                                 *
***************************************************************
⟨search..., read TEACHERS into IREC, EOFSW, SVREAD return 112⟩
        ************************************************************
        *       FORMAT - Format a single detail line.                *
        ************************************************************
        FORMAT  EQU   *
                ST    R10,SVFORM
                MVC   OTID,ITID                Move teacher ID Nbr to output
                MVC   OTNAME,ITNAME            Move teacher Name to output
                MVC   OTDEG,ITDEG              Move highest degree to output
                MVC   OTTEN,ITTEN              Move tenure to output
                MVC   OTPHONE,ITPHONE          Move phone nbr to output
                MVC   OCRLF,WCRLF              PC/370 ONLY - end line w/ CR/LF
                L     R10,SVFORM
                BR    R10
***************************************************************
*       CHKLNS - Check lines printed. Full page?              *
***************************************************************
⟨search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS 184⟩
***************************************************************
*       WRITE - Write a single detail line.                           *
***************************************************************
⟨search..., write REPORT from OREC, LNS++, SVWRITE return 185⟩
        ************************************************************
        *       WRAPUP - Those things which happen one time only,              *
        *                 after all records have been processed.               *
        ************************************************************
```

```
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE TEACHERS
         CLOSE REPORT
         WTO   'TEACH8A ... Teacher list on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
```

⟨*TEACHERS DCB, EODAD=ATEND* 26c⟩

```
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*************************************************************
*        RETURN ADDRESSES                                   *
*************************************************************
```

⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩

```
SVCHKLNS DC    F'0'                        CHKLNS
         *****************************************************************
         *        Miscellaneous field definitions                        *
         *****************************************************************
         WCRLF    DC    X'0D25'                    PC/370 ONLY - EBCDIC CR/LF
         EOFSW    DC    CL1'N'                     End of file? (Y/N)
         PGS      DC    PL2'0'                     Nbr of pages printed.
         LNS      DC    PL2'3'                     Lines printed on this page.
         MAXLNS   DC    PL2'3'                     Max nbr lines per page.
         *                                         My line counts exclude hdgs.
         *****************************************************************
         *        Input record definition                                *
         *****************************************************************
```

⟨*TEACHER IREC, registro de entrada* 44⟩

```
         *****************************************************************
         *        Output (line) definition                               *
         *****************************************************************
         OREC     DS    0CL62
         OTID     DS    CL3                        Teacher ID nbr
                  DC    CL3' '
         OTNAME   DS    CL15                       Teacher name
                  DC    CL3' '
         OTDEG    DS    CL4                        Highest degree
                  DC    CL3' '
         OTTEN    DS    CL1                        Tenured?
```

```
                DC    CL3' '
          OTPHONE DS    CL4                        Phone nbr
                  DC    CL21' '
          OCRLF   DS    CL2                        PC/370 only - CR/LF
          *
          *       Headings definitions
          *
          FORMFEED DS   0CL62                      PC/370 only
                   DC   X'0C'                      EBCDIC formfeed
                   DC   CL59' '
                   DC   X'0D25'                    EBCDIC CR/LF
          HD1      DS   0CL62
                   DC   CL40'                  LIST OF TEACHERS                    Page
          HDPGS    DC   CL3' '
                   DC   CL17' '
                   DC   XL2'0D25'
          HD2      DS   0CL62
                   DC   CL60' '
                   DC   XL2'0D25'
HD3     DS    0CL62
              DC    CL40'ID#      Name        Degr  Ten Phone'
              DC    CL20' '
              DC    XL2'0D25'
       HD4    DS    0CL62
              DC    CL40'--
              DC    CL20' '
              DC    XL2'0D25'
              END   BEGIN
```

La ejecución y salida se muestran también. Mensajes de la OMC son utilizadas, como antes, para indicar la progreso del programa y para recordar a los usuarios dónde encontrar la salida.

De comandos de DOS TIPO 'se utiliza para ver la salida. Tenga en cuenta que un carácter de avance de página, cuando se muestra con el comando TYPE, aparece como una "hembra" símbolo. El comando COPY puede utilizarse para imprimir la salida. (PRN es un nombre de dispositivo de DOS sentido de la impresora.)

```
A:\MIN>teach8a
       TEACH8A ... Begin execution
       TEACH8A ... Teacher list on REPORT.TXT
       TEACH8A ... Normal end of program
A:\MIN>type report.txt
       o             LIST OF TEACHERS        Page 001
       ID#         Name           Degr Ten  Phone
       ---   ---------------     ---- ---  -----
       732   BENSON, E.T.        PHD   N   5156
       218   HINCKLEY, G.B.      MBA   N   5509
       854   KIMBALL, S.W.       PHD   Y   5594
       o             LIST OF TEACHERS        Page 002
       ID#         Name           Degr Ten  Phone
       ---   ---------------     ---- ---  -----
       626   YOUNG, B.           MBA   Y   5664
       574   SMITH, J.           MS    Y   5320
A:\MIN>copy report.txt prn
        1 file(s) copied
A:\MIN>
```

8.2.4. Ahorrar papel

Para ahorrar papel cuando se prueba, considere la siguiente modificación:

```
FORMFEED DS 0CL62   PC/370 only
*        DC X'0C'   EBCDIC formfeed
*        DC CL59' '
         DC 60C'_'  For testing...
         DC X'0D25' EBCDIC CR/LF
```

8.3. Ejercicios

1. Verdadero o falso.

   a) El campo de número de páginas (PGS) debe ser inicializado a 1. F T

   b) El campo de número de páginas (PGS) se incrementa antes de pasar a la F T campo correspondiente en las partidas.

c) El campo de número de páginas en las partidas debe ser definido como un campo de carga. F T

d) El símbolo de avance de página es el ASCII 12. F T

e) El símbolo de avance de papel aparecen como el signo femenino cuando se ve en el F T pantalla.

f) EBCDIC 12 es reconocido como un avance de página en las impresoras de la mayoría de los mainframe. F T

g) En la rutina HDGS se muestra en este capítulo, el número de línea (LNS) se pone a cero F T porque "las partidas son libres".

h) Comando COPY de DOS 'puede ser usado para enviar el informe a la impresora. F T

i) La línea de FORMFEED debe ser la misma longitud que cualquier línea de partida de otros. F T

j) La rutina de CHKLNS se invoca por primera vez en la rutina de instalación. F T

k) El recuento de línea (LNS) debe ser inicializado a cero. F T

l) CHKLNS invocará HDGS LNS si es inferior a MAXLNS. F T

m) LNS se incrementa después de PUT en la rutina de WRITE. F T

2. Mostrar los cambios necesarios para el programa de ejemplo, TEACH8A.MLC, si el informe se está espera que más de 999 páginas de largo.

3. Mostrar los cambios necesarios para el programa de ejemplo, TEACH8A.MLC, si, a diferencia del autor, prefiere las partidas se incluyen en la cuenta en línea.

4. Muchas de las impresoras de la PC original no reconocer ASCII 12 como un carácter de avance. La mayoría de estas impresoras ni siquiera tienen un botón de avance de página. Para avanzar a la parte superior de la Página de estas impresoras, es necesario escribir un número suficiente de líneas en blanco, como 66-LNS. Modificar la HDGS rutina para utilizar este método en lugar de FORMFEED como se muestra. Para ahorrar papel, usted puede asumir cada página es de 22 líneas de largo, lo cual permite simular las tres páginas de la producción (y, por tanto, tres conjuntos de líneas) en una sola página.

5. El ejercicio 6 del capítulo 2, le pidió que elaborar una lista con formato de los registros del estudiante archivo. Modificar ese programa para la primera línea de partida incluirá un número de página. Imprimir una máximo de cinco líneas de detalle en cada página. Además, todas las líneas de detalle de impresión a doble espacio.

6. El ejercicio 7 del capítulo 2, le pidió que elaborar una lista con formato de los registros en el grado de archivo. Modificar ese programa para la primera

línea de partida incluirá un número de página. Imprimir una máximo de cinco líneas de detalle en cada página. Además, todas las líneas de detalle de impresión a doble espacio.

7. El ejercicio 8 del capítulo 2, le pidió que elaborar una lista con formato de los registros en el curso de archivo. Modificar ese programa para la primera línea de partida incluirá un número de página. Imprimir una máximo de cinco líneas de detalle en cada página. Además, todas las líneas de detalle de impresión a doble espacio.

8. El ejercicio 16 del capítulo 3 te pidió que elaborar una lista con formato de todos los grados de Inglés clases para el año escolar 1992-93 (los semestres F92 y W93 solamente.) Modificar ese programa tan primero la línea de partida incluirá un número de página. Imprimir un máximo de cuatro líneas por cada detalle página. Además, todas las líneas de detalle de impresión a doble espacio.

9. No es raro para incluir a los pies de página, así como las cabeceras, en un informe. Modificar uno de los por encima de los ejercicios para imprimir un pie de página, con el número de página correspondiente, en la parte inferior de cada uno de página. Este pie de página deben aparecer, además de la cabecera.

Capítulo 9

La Instrucción Editar

Objetivos Al término de este capítulo usted será capaz de:

- Definir una máscara de entrada para suprimir ceros a la izquierda,
- Definir una máscara de edición para incluir una coma en su caso,
- Definir una máscara de entrada para incluir un punto decimal en su caso,
- Definir una máscara de entrada para incluir un signo,
- Definir una máscara de imprimir CR PP de crédito o de débito,
- Definir una máscara de edición para la protección de cheques, y
- Usar la instrucción de editar con una máscara para imprimir un número de envasado en el formato deseado.

Introducción En el capítulo siete se introdujo Bolsas de operaciones de la aritmética decimal. En ese capítulo preparó dos informes para las Industrias Dindón: la Crónica de ventas e inventario discrepancias informes. Esos informes apareció como sigue:

```
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                        COGSWORTH INDUSTRIES
  Sales Recap
  Product        Calif        Ill         Utah        Wisc       TOTAL
  ----------     -----       -----       -----       -----       -----
  GIZMOS          020         030         020         020         090
  WIDGETS         015         010         010         002         037
  JUNQUE          025         015         015         018         073
------------------------------------------------------------------------
            1         2         3         4         5         6
  123456789012345678901234567890123456789012345678901234567890123456789012345
```

```
                    COGSWORTH INDUSTRIES
                 Inventory Discrepancies Report
Product      Begin + Purch - Sales = Expect          Actual       Result
----------   -----     -----   -----      ------     ------     ----------
GIZMOS        017       099     090        026       023        003 short
WIDGETS       022       034     037        019       019
JUNQUE        030       052     073        009       010        001 over
003 records processed.
001 indicate shortage.
001 indicate overage.
```

Se utilizó la operación UNPK para mover los resultados de una operación aritmética a un campo de salida, y la se utilizó la operación MVZ para eliminar el signo. (Hemos visto que un número con signo que a veces de impresión como una carta.) Recuerde que en nuestro debate en ese capítulo que hay dos problemas con esta técnica:

- ceros a la izquierda no se suprimen, y
- el signo (positivo o negativo) se pierde.

En este capítulo se presenta la edición de instrucción, que nos permitirá eliminar ceros a la izquierda e imprimir un signo, así como para realizar todas las operaciones de formato que se ha visto en otros idiomas, tales como la inserción de una coma, la protección de cheques, etc

9.0.1. Problemas con UNPK y MVZ Revistited

Supongamos que se nos da FLDA define como PL4, con un valor de 123.456:

```
FLDA DC PL4'123456 '01 23 45 6C
```

Para imprimir este campo, podríamos usar UNPK y MVZ de la siguiente manera:

```
WK7        UNPK WK7, FLDA F0 F1 F2 F3 F4 F5 C6        MVZ WK7 6 (1), F0 = X'F0 'F1 F2 F
```

Pero ¿y si FLDA había sido negativo? Nos han perdido el signo, como sigue:

```
FLDA DC PL4'-123456-01 23 45 6D
```

En ambos casos, WK7 tiene el mismo valor después de la UNPK y MVZ. Del mismo modo, ¿qué pasa si FLDA representada en dólares y centavos, es decir, lo que si quería ver a 1234,56? Teniendo en cuenta las definiciones de campo:

```
IMPORTE DS 0CL8        DÓLARES DS CL5        DECIMAL DS CL1        CENTAVOS DS CL2 ...
```

podríamos hacer lo siguiente:

```
CANTIDAD        DÓLARES MVC, WK7 F0 F1 F2 F3 F4        MVI DECIMAL, C '. " F0 F1 F2 F3
```

Pero todavía tenemos el problema con la señal. ¿Y qué acerca de la supresión de ceros a la izquierda? Y insertar una coma? Para lograr estas cosas en el BAL, usamos la edición (ED) la instrucción. Esto le da nos las mismas capacidades en otros idiomas. Por ejemplo, en COBOL que el código:

```
WORKING-STORAGE SECTION.
01 MISC.
    05 FLDA     PIC S9(5)V99 COMP-3 VALUE +1234.56.
    05 PRTA     PIC ZZ,ZZZ.99.
PROCEDURE DIVISION.
    MOVE FLDA TO PRTA.
```

... y en BASIC que el código:

```
10 LET FLDA = 1234.56
20 LET MASK$ = "##,###.##"
30 PRINT USING MASK$; FLDA
```

En ambos casos, el resultado es "B1, 234,56".

9.0.2. Definición de la máscara de edición

Hay tres pasos para la edición de un número:

1. Defina una máscara de edición,
2. Mover la máscara para el campo de destino, y
3. Editar un número de envasados en ese campo de destino.

Para editar un número, debe definir una máscara que indica el formato deseado. El primer byte de la máscara de contendrá siempre un carácter de relleno que indica el carácter con que ceros a la izquierda se reemplazado. Esto suele ser un blanco (X'40 '), sino que puede ser un asterisco (X'5C') para comprobar la protección. El carácter de relleno es seguido por muchos como X'20 como hay dígitos en la Bolsas de número que se está editando. Seguimos con el ejemplo anterior. Teniendo en cuenta FLDA define como PL4, con una valor de 123.456:

```
FLDA DC PL4'123456 '01 23 45 6C
```

Queremos imprimir este campo con ceros a la izquierda suprimida. Teniendo en cuenta la definición de campo siguientes:

```
MASK DC X'4020202020202020'
```

tenemos el siguiente código:

```
WK8        MVC WK8, MÁSCARA 40 20 20 20 20 20 20 20        ED WK8, FLDA 40 40 F1 F2 F3
```

(Recuerde que cuando se utiliza una etiqueta de la forma WKN, estoy dando a entender que un campo de trabajo de n bytes de longitud se ha definido, por ejemplo, WK8 CL8 DS.)

Observe que la máscara tiene un carácter de relleno (X'40 ') que indica que todos los ceros a la izquierda se sustituye con espacios en blanco. Además, hay siete X'20 ', porque un campo definido como PL4 tiene siete dígitos. Por lo tanto la máscara es un total de ocho bytes de longitud. Por supuesto, podría haber codificado el siguiente en su lugar:

```
MVC WK8,=X'4020202020202020'
ED  WK8,FLDA
```

El literal (= X'4020202020202020 ') parecería entonces después de la LTORG en la lista de montaje. De Por supuesto, si FLDA tenía un valor de cero, sería WK8 todos los espacios. Por ejemplo, FLDB dado y MASK2 define como sigue:

```
FLDB  DC PL3'0 '
MASK2 DC X'402020202020 '
```

...tenemos el siguiente código:

```
WK6       MVC WK6, MASK2 40 20 20 20 20 20       ED WK6, FLDB 40 40 40 40 40 40 bbbbbb
```

Tenga en cuenta que la máscara debe ser trasladado a la zona de salida cada vez que un número será editado por la edición de instrucciones destruye la máscara. Por último, la edición de instrucción es una instrucción de tipo SS-similar de la MVC, es decir, la longitud de la secuencia está determinada por la longitud del campo de recepción solamente. La edición continúa durante tantos dígitos como se representan en el campo de recepción.

intenta Dada X DC PL3'1234 y MASK DC XL6'402020202020 ', muestran el valor de WK6 o WK4 después de cada de los siguientes:

```
1. UNPK WK6, X
2. UNPK WK6, X     MVZ WK6 +5 (1), = X'F0 '
3. MVC WK6, MÁSCARA
   ED WK6, X
4. MVC WK4, MASK ¡Cuidado!
   ED WK4, X +1
```

9.0.3. Indicando Importancia

Si queremos detener la supresión de ceros a la izquierda, con el fin de forzar al menos un cero a la impresión, que sustituir a un X'20 único "con un X'21 '. El X'21 'es el último dígito que será sustituido por el relleno carácter. En otras palabras, los ceros a la derecha de la X'21 'se imprimirá. Por ejemplo, dada:

```
FLDB DC PL3'0 '
MASK3 DC X'402020202120 '
```

...tenemos el siguiente código:

```
WK6       MVC WK6, MASK3 40 20 20 20 21 20       ED WK6, FLDB 40 40 40 40 40 F0 bbbbb0
```

Nota: el número total de X'20 y X'21 siempre será extraño!

intenta Habida cuenta de escribir la instrucción (s) para mover PJS a ...

```
         PJS PL3' DC-49 '
5. ... WK6 de tal manera que se C'bbb049.                    WK6
6. ... WK6 de tal manera que WK6 se C'bb0049.
7. ... WK4 tal que WK4 se C'bb49.                  WK4 se C'004R. (Sugerencia: = C
8. ... De tal manera que WK4
```

9.0.4. Impresión Decimal Puntos

¿Qué pasa con los puntos decimales? Nosotros ni siquiera mencionó los puntos decimales en nuestra discusión de concentrado de la aritmética decimal. Eso es porque no hay en el BAL, es decir, toda la aritmética en BAL es un número entero aritmética. Depende de usted como el programador para realizar un seguimiento del lugar en el decimal implícita es. Esto complica cualquier aritmética (como veremos en un capítulo posterior) y el formato. Para imprimir un decimal punto, se añade un punto (X'4B ') en la posición adecuada dentro de la máscara. Por ejemplo, dada nuestros anteriores definiciones de FLDA y FLDB:

```
MVC WK9,=X'4020202020214B2020'
ED  WK9,FLDA
MVC WK7,=X'402020214B2020'
ED  WK7,FLDB
```

Tenga en cuenta que las máscaras se están más ....

Cómo va a mostrar ceros cuando la impresión de dólares y centavos, es una cuestión de preferencia personal. Para ejemplo, usted puede preferir que la parte en blanco de dólares (como en WK7 arriba), o puede preferir mostrar un cero en la parte de dólares. En este último caso, la máscara sería cambiado con la X'21 ' descendió una posición a la izquierda. Por ejemplo:

```
MVC WK7, = X'402021204B2020'
ED WK7, FLDB
```

... da WK7 igual a C'bbb0 .00 '.

intenta Habida cuenta de escribir la instrucción (s) para mover H ...

```
H DC PL3'6 ' 9. ... WK7 tal que WK7 se C'bbbbb .6.
10. ... WK7 tal que WK7 se C'bbb00 .6.
11. ... WK7 tal que WK7 se C'bbb0 .06 '.
```

Impresión de comas De la misma manera que los períodos de añadir a la salida, podemos añadir una coma a la salida. Para imprimir una coma, añadir una coma (X'6B ') en la posición adecuada (s) dentro de la máscara. Continuando con nuestro ejemplo anterior:

```
MVC WK10,=X'4020206B2020214B2020'
ED  WK10,FLDA
```

... da WK10 igual a C'bb1, 234,56. Tenga en cuenta que nuestro campo de destino y la máscara han pasado de ocho bytes a diez bytes! Tenga en cuenta también que el número total de X'20 y X'21 todavía es impar (siete): una para cada dígito en FLD.

intenta Habida cuenta de escribir la instrucción (s) para mover W a ...

```
       W DC PL5'6301982 '
12. ... WK10 tal que WK10 se C'bbb6301982.
13. ... WK12 tal que WK12 se C'bbb6, 301.982.
14. ... WK12 tal que WK12 se C'bbb63, 019,82.
15. ... WK12 tal que WK12 se C'bb063, 019,82.
16. ... WK7 tal que WK7 se C'bb1, 982 '. (este es un tanto complicado!)
```

Habida cuenta de escribir la instrucción (s) para mover C a ...

```
       C DC PL4'72384 '
17. ... WK9 tal que WK9 se C'bbb72, 384 '.
18. ... WK10 tal que WK10 se C'bbb7, 238,4.
```

9.0.5. Impresión de la Señal

```
Considere el siguiente ejemplo. Teniendo en cuenta:
POS   DC PL3'+123'         A positive number
NEG   DC PL3'-123'         A negative number
MASK4 DC X'402021204B2020'
```

si el código de instrucciones siguientes obtener los mismos resultados:

```
WK7        MVC WK7, mask4 40 20 21 20 4B 20 20        ED WK7, POS 40 40 40 F1 4B F2 F3
```

Vemos que tanto el POS y NEG serán impresos como C'bbb1 .23 '. La instrucción ED elimina el signo. Para corregir este problema, podemos añadir un guión (X'60 ') al final de la máscara. Este guión se sustituye por el carácter de relleno, si el número no es negativo. Por ejemplo, dada:

```
MASK5 DC X'402021204B202060'
```

...si el código de las siguientes instrucciones que recibe diferentes (correctos):

```
WK8        MVC WK8, MASK5 40 20 21 20 4B 20 20 60        ED WK8, POS 40 40 40 F1 4B F2
```

Vemos que las OP se imprimirá como C'bbb1 .23 b », mientras que NEG se imprimirá como C'bbb1 .23 - '. (La instrucción EDMK se puede utilizar para imprimir un signo inicial en comparación con un final signo: será discutido en un capítulo posterior.)

Tenga en cuenta que cuando he cambiado la máscara, que tenía que aumentar la longitud del campo de recepción (que solía WK8 en lugar de WK7). Si no lo hace es la fuente de muchos errores de programación para el comienzo de BAL los programadores.

Recuerde: la longitud del campo de recepción debe ser la misma que la longitud de la la máscara, y que la máscara debe tener como muchos X'20 's y X'21' s ya que hay dígitos en el campo de Bolsas de que está siendo editada!

intenta Habida cuenta de escribir la instrucción (s) para mover E de la ...

```
        E PL2' DC-4 '
19. ... WK5 de tal manera que se C'bbb4-'.                    WK5
20. ... WK6 de tal manera que WK6 se C'bb .04 - '.
```

9.0.6. Impresión CR (crédito) o DB (de débito)

Del mismo modo, si estamos trabajando en una aplicación de contabilidad, podemos añadir CR (X'C3D9 ') o DB (X'C4C2 ') a nuestra máscara para indicar un crédito o débito, respectivamente. El CR (o PP) se van a imprimir, si el número es negativo, de lo contrario se sustituye por el carácter de relleno. Por ejemplo, dada:

```
POS DC PL3'+123'
NEG DC PL3'-123'
CR  DC X'402021204B2020C3D9'
DB  DC X'402021204B2020C4C2'
```

Las instrucciones siguientes le dan los resultados indicados:

```
WK9        D9 MVC WK9, CR 40 20 21 20 4B 20 20 C3        ED WK9, POS 40 40 40 F1 4B F2
```

intenta Dado

```
PL2’ B DC-38 ’,
```

escriba la instrucción (s) para mover B de la ...

```
21. ... WK6 de tal manera que se C’bb38DB.                    WK6
22. ... WK7 tal que WK7 se C’bb .38 CR.
23. ... WK7 tal que WK7 se C’bb3 .8 BB ’.
```

9.0.7. Hora de Protección

Hasta ahora hemos utilizado un blanco como el carácter de relleno en todas nuestras máscaras. Para la protección de comprobación, Puede utilizar un asterisco (X’5C ’). Por ejemplo, dada nuestra anterior definición de FLDA y FLDB, y el siguientes definiciones para chka y CHKB:

```
Chka DC X’5C20206B2021204B2020 ’
CHKB DC X’5C2021204B2020 ’
```

...Las instrucciones siguientes le dan los resultados indicados:

```
WK10        MVC WK10, chka 5C 20 20 6B 20 21 20 4B 20 20        ED WK10, FLDA 5C 5C F1
```

intenta Determinada, debido

```
DC PL4’6591 ’
```

, escriba la instrucción (s) para moverse a causa de ...

```
24. ... WK10 tal que WK10 será C’***** 6591.
25. ... WK9 tal que WK9 será C’**** 6591.
26. ... WK11 tal que WK11 será C’***** 65.91b.
```

9.0.8. La documentación de la máscara de edición

Para simplificar el mantenimiento del programa, me gusta "documento", las máscaras de impresión mediante la presentación del representación de caracteres de los campos hexadecimal utilizado en la máscara. Yo uso una B para indicar un espacio en blanco como la principales caracter de relleno, un Z para indicar un dígito en ceros a la izquierda será suprimida, y un 9 para indicar un dígito en ceros a la izquierda no será suprimida. Por ejemplo, me gustaría de las máscaras de impresión de documentos que hemos utilizado hasta ahora como:

```
MASK  DC X'4020202020202020'     BZZZZZZZ
MASK2 DC X'402020202020'         BZZZZZ
MASK3 DC X'402020202120'         BZZZZ9
MASK4 DC X'402021204B2020'       BZZ9.99
MASK5 DC X'402021204B202060'     BZZ9.99-
CR    DC X'402021204B2020C3D9'   BZZ9.99CR
DB    DC X'402021204B2020C4C2'   BZZ9.99DB
CHKA  DC X'5C20206B2021204B2020' ***,**9.99
CHKB  DC X'5C2021204B2020'       ***9.99
```

Con la excepción de la B líderes, los programadores de COBOL reconocerá esta notación que se utiliza en las cláusulas COBOL PIC's. De ahora en adelante, voy a utilizar esta misma notación en las especificaciones de impresión para todos los ejercicios.

9.1. Muestra del Programa

El siguiente programa, EDITS.MLC, ilustra los ejemplos que hemos utilizado hasta ahora. La OMC (escribir al operador) macro se usa para mostrar los resultados de las ediciones. Además de ilustrar el uso de de la instrucción ED, espero que este programa muestra cómo puede utilizar el comando de la OMC a la experimentar con esta y otras indicaciones como intentar aprender BAL.

202 ⟨ff 40⟩+≡ ◁101 207▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: EDITS9.MLC                                   *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Demonstrate the edit instruction by          *
*                  implementing examples shown in chapter 9.    *
*****************************************************************
         START 0
BEGIN    BEGIN
*****************************************************************
         WTO   'SEE PAGE 9.3'
*****************************************************************
         MVC   WK8,MASK
         ED    WK8,FLDA
         WTO   WK8
         MVC   WK8,=X'4020202020202020'
         ED    WK8,FLDA
         WTO   WK8
*****************************************************************
         WTO   'SEE PAGE 9.4'
*****************************************************************
```

```
         MVC   WK6,MASK2
         ED    WK6,FLDB
         WTO   WK6
         MVC   WK6,MASK3
         ED    WK6,FLDB
         WTO   WK6
*****************************************************************
         WTO   'SEE PAGE 9.5'
*****************************************************************
         MVC   WK9,=X'4020202020214B2020'
         ED    WK9,FLDA
         WTO   WK9
         MVC   WK7,=X'402020214B2020'
         ED    WK7,FLDB
         WTO   WK7
         MVC   WK7,=X'402021204B2020'
         ED    WK7,FLDB
         WTO   WK7
*****************************************************************
         WTO   'SEE PAGE 9.6'
*****************************************************************
         MVC   WK10,=X'4020206B2020214B2020'
         ED    WK10,FLDA
         WTO   WK10
         MVC   WK7,MASK4
         ED    WK7,POS
         WTO   WK7
         MVC   WK7,MASK4
         ED    WK7,NEG
         WTO   WK7
*****************************************************************
         WTO   'SEE PAGE 9.7'
*****************************************************************
         MVC   WK8,MASK5
         ED    WK8,POS
         WTO   WK8
         MVC   WK8,MASK5
         ED    WK8,NEG
         WTO   WK8
*****************************************************************
         WTO   'SEE PAGE 9.8'
*****************************************************************
         MVC   WK9,CR
         ED    WK9,POS
         WTO   WK9
         MVC   WK9,CR
```

```
               ED    WK9,NEG
               WTO   WK9
               MVC   WK9,DB
               ED    WK9,POS
               WTO   WK9
               MVC   WK9,DB
               ED    WK9,NEG
               WTO   WK9
               MVC   WK10,CHKA
               ED    WK10,FLDA
               WTO   WK10
          MVC    WK7,CHKB
          ED     WK7,FLDB
          WTO    WK7
     ****************************************************************
          WTO    'ALL DONE...'
     ****************************************************************
```

⟨*Epilogo* 26a⟩

```
*
     *         Other field definitions
     *
     WK6       DS     CL6
     WK7       DS     CL7
     WK8       DS     CL8
     WK9       DS     CL9
     WK10      DS     CL10
     FLDA      DC     PL4'123456'
     FLDB      DC     PL3'0'
     POS       DC     PL3'+123'
     NEG       DC     PL3'-123'
     MASK      DC     X'4020202020202020'      BZZZZZZZ
     MASK2     DC     X'402020202020'          BZZZZZ
     MASK3     DC     X'402020202120'          BZZZZ9
     MASK4     DC     X'402021204B2020'        BZZ9.99
     MASK5     DC     X'402021204B202060'      BZZ9.99-
     CR        DC     X'402021204B2020C3D9'    BZZ9.99CR
     DB        DC     X'402021204B2020C4C2'    BZZ9.99DB
     CHKA      DC     X'5C20206B2021204B2020'  ***,**9.99
     CHKB      DC     X'5C2021204B2020'        ***9.99
               END    BEGIN
```

```
A:\MIN>edits9
SEE PAGE 9.3
  123456
  123456
SEE PAGE 9.4
     0
SEE PAGE 9.5
  1234.56
    .00
   0.00
SEE PAGE 9.6
  1,234.56
   1.23
   1.23
SEE PAGE 9.7
   1.23
   1.23-
SEE PAGE 9.8
   1.23
   1.23CR
   1.23
   1.23DB
**1,234.56
***0.00
ALL DONE...
```

9.2. Informes Dindón Revisited:

La Crónica de Ventas Volvemos ahora a los problemas de programación que motivaron esta discusión: Ventas de Dindón Recapitulación y discrepancias de inventario informes.

El diseño de impresión siguientes para la Crónica de ventas ha sido modificado para utilizar la salida de editado. Recuerde que de un campo de diseño de BZZ9 corresponde a una máscara de X'40202120. Estos campos importe anteriormente define como tres bytes de longitud. Ahora son cuatro bytes de longitud por lo que la definición de registro de salida deberá ser modificado en consecuencia.

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                  COGSWORTH INDUSTRIES
Sales Recap
Product       Calif      Ill      Utah    Wisc     TOTAL
```

```
----------    -----     -----   -----    -----     -----
XXXXXXXXXX    BZZ9      BZZ9    BZZ9     BZZ9      BZZ9
XXXXXXXXXX    BZZ9      BZZ9    BZZ9     BZZ9      BZZ9
XXXXXXXXXX    BZZ9      BZZ9    BZZ9     BZZ9      BZZ9
BZZ9 records processed.
```

Anteriormente hemos utilizado MVC (sólo) para mover las ventas por el Estado a sus respectivos campos de salida. Nosotros Bolsas de los campos sólo por lo que podrían llegar a ellos. Ahora, estos campos envasados se utilizarán para que podamos mostrar la salida con ceros a la izquierda suprimida:

```
PACK WCALIF,ICALIF Each product's sales must
PACK WILL,IILL      be packed so they can be
PACK WUTAH,IUTAH     added to total for this
PACK WWISC,IWISC      product...
MVC  OCALIF,WMASK
ED   OCALIF,WCALIF
MVC  OILL,WMASK
ED   OILL,WILL
MVC  OUTAH,WMASK
ED   OUTAH,WUTAH
MVC  OWISC,WMASK
ED   OWISC,WWISC
```

donde:

```
WMASK DC X'40202120' BZZ9
```

Del mismo modo, el número de registros se muestran como sigue:

```
MVC OREC(23),=CL23'BZZ9 records processed.'
MVC OREC(4),WMASK
ED  OREC(4),#IN        Count
```

El programa completo y su producción de seguir.

206 ⟨*search..., setup ascii INVENTRY and REPORT, HDGS, prime READ, SVSETUP return* 206⟩≡ (157 207 2

```
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     INVENTRY+10,X'08'         PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN INVENTRY
         OPEN REPORT
         BAL    R10,HDGS
         BAL    R10,READ
         L      R10,SVSETUP
         BR     R10
```

207 ⟨*ff* 40⟩+≡ ◁202 249▷

```
                PRINT NOGEN
*****************************************************************
*       FILENAME: COGS9A.MLC                                    *
*       AUTHOR : Bill Qualls                                    *
*       SYSTEM : PC/370 R4.2                                    *
*       REMARKS : Produce report for COGSWORTH INDUSTRIES       *
*                   showing sales by state.                     *
*                   Modify COGS7A.MLC to use ED instruction.    *
*****************************************************************
                START 0
                REGS
BEGIN           BEGIN
                WTO     'COGS9A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
                WTO     'COGS9A ... Normal end of program'
                RETURN
*************************************************************
*       SETUP - Those things which happen one time only,    *
*               before any records are processed.           *
*************************************************************
⟨search..., setup ascii INVENTRY and REPORT, HDGS, prime READ, SVSETUP return 206⟩
*****************************************************************
*       HDGS - Print headings.                                  *
*****************************************************************
HDGS            EQU     *
                ST      R10,SVHDGS
                PUT     REPORT,HD1
                PUT     REPORT,HD2
                PUT     REPORT,HD3
                PUT     REPORT,HD4
                PUT     REPORT,HD5
                L       R10,SVHDGS
                BR      R10
*****************************************************************
*       PROCESS - Those things which happen once per record.    *
*****************************************************************
PROCESS EQU     *
                ST      R10,SVPROC
                BAL     R10,FORMAT
                BAL     R10,WRITE
                BAL     R10,READ
                L       R10,SVPROC
                BR      R10
*************************************************************
*       READ - Read a record.                               *
```

```
*************************************************************
⟨search..., read INVENTRY into IREC, EOFSW,IN++ SVREAD return 156⟩
          *************************************************************
          *        FORMAT - Format a single detail line.              *
          *************************************************************
          FORMAT   EQU   *
                   ST    R10,SVFORM
                   MVC   OREC,BLANKS
                   MVC   ODESC,IDESC
                   PACK WCALIF,ICALIF              Each product's sales must
                   PACK WILL,IILL                    be packed so they can be
                   PACK WUTAH,IUTAH                    added to total for this
                   PACK WWISC,IWISC                     product...
                   MVC   OCALIF,WMASK
                   ED    OCALIF,WCALIF
                   MVC   OILL,WMASK
                   ED    OILL,WILL
                   MVC   OUTAH,WMASK
                   ED    OUTAH,WUTAH
                   MVC   OWISC,WMASK
                   ED    OWISC,WWISC
                   ZAP   WTOTAL,=P'0'              Initialize the total to zero
                   AP    WTOTAL,WCALIF               and start adding...
                   AP    WTOTAL,WILL
                   AP    WTOTAL,WUTAH
                   AP    WTOTAL,WWISC
                   MVC   OTOTAL,WMASK
                   ED    OTOTAL,WTOTAL
                   MVC   OCRLF,WCRLF               PC/370 only.
                   L     R10,SVFORM
                   BR    R10
*************************************************************
*        WRITE - Write a single detail line.                *
*************************************************************
⟨search..., write REPORT from OREC, no line counting, SVWRITE return 106⟩
          *************************************************************
          *        WRAPUP - Those things which happen one time only,  *
          *                 after all records have been processed.    *
          *************************************************************
          WRAPUP   EQU   *
                   ST    R10,SVWRAP
                   MVC   OREC,BLANKS
                   MVC   OCRLF,WCRLF               PC/370 only.
                   BAL   R10,WRITE                 Skip a line.
                   MVC   OREC(23),=CL23'BZZ9 records processed.'
                   MVC   OREC(4),WMASK
```

```
         ED    OREC(4),#IN              Count
         BAL   R10,WRITE
         CLOSE INVENTRY
         CLOSE REPORT
         WTO   'COGS9A ... Sales recap on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*************************************************************
*        RETURN ADDRESSES                                   *
*************************************************************
```

⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩

```
*****************************************************************
*        Miscellaneous field definitions                                        *
*****************************************************************
WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                   End of file? (Y/N)
BLANKS   DC    CL62' '
WCALIF   DC    PL2'0'                   Units sold in Calif
WILL     DC    PL2'0'                   Units sold in Illinois
WUTAH    DC    PL2'0'                   Units sold in Utah
WWISC    DC    PL2'0'                   Units sold in Wisconsin
WTOTAL   DC    PL2'0'                   Units sold in all states
#IN      DC    PL2'0'                   Input record count
WMASK    DC    X'40202120'              BZZ9
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL41             1-41 Inventory record
IDESC    DS    CL10              1-10 Product description
ICALIF   DS    CL3              11-13 Units sold in Calif
IILL     DS    CL3              14-16 Units sold in Illinois
IUTAH    DS    CL3              17-19 Units sold in Utah
IWISC    DS    CL3              20-22 Units sold in Wisconsin
IBEGIN   DS    CL3              23-25 Beginning inventory
IPURCH   DS    CL3              26-28 Purchases throughout year
```

```
*
IQOH     DS    CL3             29-31 Actual quantity on hand
ICOST    DS    CL4             32-35 Cost (each) 99V99
ISELL    DS    CL4             36-39 Sell for (each) 99V99
ICRLF    DS    CL2             40-41 PC/370 only - CR/LF
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62            1-62
ODESC    DS    CL10             1-10 Product description
         DS    CL4             11-14
OCALIF   DS    CL4             15-18 Units sold in Calif
         DS    CL5             19-23
OILL     DS    CL4             24-27 Units sold in Illinois
         DS    CL5             28-32
OUTAH    DS    CL4             33-36 Units sold in Utah
         DS    CL5             37-41
OWISC    DS    CL4             42-45 Units sold in Wisconsin
         DS    CL5             46-50
OTOTAL   DS    CL4             51-54 Units sold in all states
         DS    CL6             55-60
OCRLF    DS    CL2             61-62 PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
HD1      DS    0CL62
         DC    CL40'                  COGSWORTH INDUSTRIES  '
         DC    CL20' '
         DC    XL2'0D25'
HD2      DS    0CL62
         DC    CL40'                       Sales Recap      '
         DC    CL20' '
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'Product        Calif     Ill       Utah '
         DC    CL20' Wisc      TOTAL'
         DC    XL2'0D25'
HD5      DS    0CL62
         DC    CL40'-----     ---    ---    ---  '
         DC    CL20' ---     ---'
         DC    XL2'0D25'
         END   BEGIN
```

```
A:\MIN> cogs9a
COGS9A ... Begin execution
COGS9A ... Sales recap on REPORT.TXT
COGS9A ... Normal end of program
A:\MIN> type report.txt
                COGSWORTH INDUSTRIES
                      Sales Recap
Product       Calif      Ill       Utah  Wisc TOTAL
----------    -----     -----     -----  ----- -----
GIZMOS          20        30        20    20    90
WIDGETS         15        10        10     2    37
JUNQUE          25        15        15    18    73
   3 records processed.
```

Informes Dindón Revisited: las discrepancias Informe de Inventario El nuevo informe de discrepancias de inventario aparecerá como sigue:

```
1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                    COGSWORTH INDUSTRIES
               Inventory Discrepancies Report
Product      Begin + Purch - Sales = Expect   Actual   Diff
----------   -----    -----   -----   ------  ------   ----
XXXXXXXXXX   BZZ9     BZZ9    BZZ9     BZZ9   BZZ9    BZZZ-
XXXXXXXXXX   BZZ9     BZZ9    BZZ9     BZZ9   BZZ9    BZZZ-
XXXXXXXXXX   BZZ9     BZZ9    BZZ9     BZZ9   BZZ9    BZZZ-
BZZ9 records processed.
BZZ9 indicate shortage.
BZZ9 indicate overage.
```

El programa completo y su producción de seguir.

211 ⟨*COGS9B.MLC* 211⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: COGS9B.MLC                                   *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce report for COGSWORTH INDUSTRIES      *
*                  showing inventory discrepancies.             *
*                  Modify COGS7B.MLC to use ED instruction.     *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'COGS9B ... Begin execution'
```

⟨*search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP* 111⟩

```
                WTO    'COGS9B ... Normal end of program'
                RETURN
*************************************************************
*        SETUP - Those things which happen one time only,   *
*                before any records are processed.          *
*************************************************************
```

⟨*search..., setup ascii INVENTRY and REPORT, HDGS, prime READ, SVSETUP return* 206⟩

```
*************************************************************
*        HDGS - Print headings.                             *
*************************************************************
HDGS    EQU   *
        ST    R10,SVHDGS
        PUT   REPORT,HD1
        PUT   REPORT,HD2
        PUT   REPORT,HD3
        PUT   REPORT,HD4
        PUT   REPORT,HD5
        L     R10,SVHDGS
        BR    R10
*************************************************************
*        PROCESS - Those things which happen once per record. *
*************************************************************
PROCESS EQU   *
        ST    R10,SVPROC
        BAL   R10,FORMAT
        BAL   R10,WRITE
        BAL   R10,READ
        L     R10,SVPROC
        BR    R10
*************************************************************
*        READ - Read a record.                              *
*************************************************************
```

⟨*search..., read INVENTRY into IREC, EOFSW,IN++ SVREAD return* 156⟩

```
*************************************************************
*        FORMAT - Format a single detail line.              *
*************************************************************
FORMAT  EQU   *
        ST    R10,SVFORM
        MVC   OREC,BLANKS
        MVC   ODESC,IDESC              Description
        PACK WBEGIN,IBEGIN             Beginning inventory
        MVC   OBEGIN,WMASK
        ED    OBEGIN,WBEGIN
        PACK WPURCH,IPURCH             Purchases
        MVC   OPURCH,WMASK
```

```
                ED    OPURCH,WPURCH
                PACK WCALIF,ICALIF              Each product's sales must
                PACK WILL,IILL                    be packed so they can be
                PACK WUTAH,IUTAH                    added to total for this
                PACK WWISC,IWISC                     product...
                ZAP   WTOTAL,=P'0'              Initialize the total to zero
                AP    WTOTAL,WCALIF               and start adding...
                AP    WTOTAL,WILL
                AP    WTOTAL,WUTAH
                AP    WTOTAL,WWISC
                MVC   OSALES,WMASK
                ED    OSALES,WTOTAL
                ZAP   WENDING,WBEGIN            Ending inventory =
                AP    WENDING,WPURCH                Beginning + Purchases
                SP    WENDING,WTOTAL                 - Sales
                MVC   OENDING,WMASK
                ED    OENDING,WENDING
                PACK WQOH,IQOH                  Actual ending inventory
                MVC   OQOH,WMASK
                ED    OQOH,WQOH
                MVC   OCRLF,WCRLF               PC/370 only.
                CP    WQOH,WENDING              Compare actual vs. expected
                BE    DODIFF
                BL    SHORT
                AP    #OVER,=P'1'               Count overages
                B     DODIFF
        SHORT   EQU   *
                AP    #SHORT,=P'1'              Count shortages
        DODIFF  EQU   *
                ZAP   WDIFF,WENDING             Difference = Expected - Actual
                SP    WDIFF,WQOH
                MVC   ODIFF,WMASK2
                ED    ODIFF,WDIFF
        FORMATX EQU   *
                L     R10,SVFORM
                BR    R10
*************************************************************
*        WRITE - Write a single detail line.                *
*************************************************************
⟨search..., write REPORT from OREC, no line counting, SVWRITE return 106⟩
          ************************************************************
          *        WRAPUP - Those things which happen one time only,     *
          *                 after all records have been processed.      *
          ************************************************************
          WRAPUP  EQU   *
                  ST    R10,SVWRAP
```

```
                MVC   OREC,BLANKS
                MVC   OCRLF,WCRLF               PC/370 only.
                BAL   R10,WRITE                 Skip a line.
                MVC   OREC(23),=CL23'BZZ9 records processed.'
                MVC   OREC(4),WMASK
                ED    OREC(4),#IN               Count all
                BAL   R10,WRITE
                MVC   OREC(23),=CL23'BZZ9 indicate shortage.'
                MVC   OREC(4),WMASK
                ED    OREC(4),#SHORT            Count shortages
                BAL   R10,WRITE
                MVC   OREC(23),=CL23'BZZ9 indicate overage. '
                MVC   OREC(4),WMASK
                ED    OREC(4),#OVER             Count overages
                BAL   R10,WRITE
                CLOSE INVENTRY
                CLOSE REPORT
                WTO   'COGS9B ... Discrepancies report on REPORT.TXT'
                L     R10,SVWRAP
                BR    R10
*****************************************************************
          *       Literals, if any, will go here                  *
          *****************************************************************
                LTORG
          *****************************************************************
          *       File definitions                                *
          *****************************************************************
          INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
                         DDNAME='COGS.DAT'
          REPORT   DCB   LRECL=67,RECFM=F,MACRF=P,
                         DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
          *****************************************************************
          *       Miscellaneous field definitions                 *
          *****************************************************************
          WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
          EOFSW    DC    CL1'N'                   End of file? (Y/N)
          BLANKS   DC    CL67' '
          WCALIF   DC    PL2'0'                   Units sold in Calif
          WILL     DC    PL2'0'                   Units sold in Illinois
          WUTAH    DC    PL2'0'                   Units sold in Utah
          WWISC    DC    PL2'0'                   Units sold in Wisconsin
          WTOTAL   DC    PL2'0'                   Units sold in all states
```

```
   WBEGIN   DC    PL2'0'                    Beginning inventory
   WPURCH   DC    PL2'0'                    Purchases
   WENDING DC     PL2'0'                    Ending inventory (expected)
   WQOH     DC    PL2'0'                    Ending inventory (actual)
   WDIFF    DC    PL2'0'                    Difference
   #IN      DC    PL2'0'                    Input record count
   #OVER    DC    PL2'0'                    Records showing overage
   #SHORT   DC    PL2'0'                    Records showing shortage
   WMASK    DC    X'40202120'               BZZ9
   WMASK2   DC    X'4020202060'             BZZZ-
   ***************************************************************
   *        Input record definition                                *
   ***************************************************************
   IREC     DS    0CL41            1-41 Inventory record
   IDESC    DS    CL10             1-10 Product description
   ICALIF   DS    CL3             11-13 Units sold in Calif
   IILL     DS    CL3             14-16 Units sold in Illinois
   IUTAH    DS    CL3             17-19 Units sold in Utah
   IWISC    DS    CL3             20-22 Units sold in Wisconsin
   IBEGIN   DS    CL3             23-25 Beginning inventory
   IPURCH   DS    CL3             26-28 Purchases throughout year
   IQOH     DS    CL3             29-31 Actual quantity on hand
   ICOST    DS    CL4             32-35 Cost (each) 99V99
   ISELL    DS    CL4             36-39 Sell for (each) 99V99
   ICRLF    DS    CL2             40-41 PC/370 only - CR/LF
   ***************************************************************
   *        Output (line) definition                               *
   ***************************************************************
OREC     DS    0CL67             1-67
ODESC    DS    CL10              1-10 Product description
         DS    CL3              11-13
OBEGIN   DS    CL4              14-17 Beginning inventory
         DS    CL4              18-21
OPURCH   DS    CL4              22-25 Purchases
         DS    CL4              26-29
OSALES   DS    CL4              30-33 Units sold
         DS    CL5              34-38
OENDING DS     CL4              39-42 Ending inventory (expected)
         DS    CL4              43-46
OQOH     DS    CL4              47-50 Ending inventory (actual)
         DS    CL4              51-54
ODIFF    DS    CL5              55-59 Difference
         DS    CL6              60-65
OCRLF    DS    CL2              66-67 PC/370 only - CR/LF
***************************************************************
*        Headings definitions                                   *
```

```
*****************************************************************
HD1      DS    0CL67
         DC    CL40'                       COGSWORTH INDUSTRIES'
         DC    CL25' '
         DC    XL2'0D25'
HD2      DS    0CL67
         DC    CL40'                Inventory Discrepancies R'
         DC    CL25'eport'
         DC    XL2'0D25'
HD3      DS    0CL67
         DC    CL65' '
         DC    XL2'0D25'
HD4      DS    0CL67
         DC    CL40'Product       Begin + Purch - Sales = Exp'
         DC    CL25'ect    Actual   Diff      '
         DC    XL2'0D25'
HD5      DS    0CL67
         DC    CL40'-----    ---   ---   ---   --'
         DC    CL25'--    ---   --      '
         DC    XL2'0D25'
         END   BEGIN
```

```
A:\MIN> cogs9b
COGS9B ... Begin execution
COGS9B ... Discrepancies report on REPORT.TXT
COGS9B ... Normal end of program
A:\MIN> type report.txt
                    COGSWORTH INDUSTRIES
               Inventory Discrepancies Report
Product       Begin + Purch - Sales = Expect   Actual Diff
----------    -----   -----   -----   ------   ------ ----
GIZMOS          17      99      90        26     23     3
WIDGETS         22      34      37        19     19
JUNQUE          30      52      73         9     10     1-
   3 records processed.
   1 indicate shortage.
   1 indicate overage.
```

9.2.1. Resumen

El número debe estar en formato decimal empaquetado para ser editada (con formato).

Para editar un número, debe definir una máscara. Esta máscara:

- La máscara debe tener la misma longitud que el campo de destino.
- El primer byte de la máscara es siempre el carácter de relleno.

- Este carácter de relleno (que sustituye a ceros a la izquierda) suele ser un espacio en blanco (X'40 '). Utilice un asterisco (X'5C ') para la protección de cheques. La máscara puede incluir una coma (X'6B ') si lo desea y, cuando proceda.
- La máscara puede incluir un punto decimal (X'4B ').
- La máscara debe incluir una X'20 'o X'21' para cada dígito en el campo de Bolsas de ser editado.
- El número total de X'20 y X'21 será siempre un número impar. La máscara puede incluir un signo final (X'60 ') o CR (X'C3D9') o DB (X'C4C2 »), que se sustituye por el carácter de relleno, si el número no es negativo.

9.3. Ejercicios

1. Verdadero o falso.

   a) Si un literal se utiliza como una máscara de edición, que literal se publicará en el LTORG. F T

   b) La longitud total de una máscara de entrada debe ser un número impar. F T

   c) La edición de instrucción, como PACK y UNPK, es una instrucción de tipo SS-que permite a F T un operador de larga duración, tanto los operandos.

   d) Un campo de salida debe ser "fresco" antes de las modificaciones posteriores, incluso si ese campo F T se definió con un DC y una máscara de edición.

   e) La X'21 'en la máscara indica el último byte en ceros a la izquierda se F T sustituye por el carácter de relleno.

   f) Si una edición termina máscara con X'C3D9 ', a continuación, los números negativos se imprimirá con CR F T y los números positivos se imprimirá con DB.

   g) Si una edición termina máscara con X'60 ', entonces los números negativos se imprimirá con un final F T signo.

   h) Una máscara de entrada debe tener un X'4B 'para cada coma. F T

   i) Si está editando el campo se define como PL3, a continuación, la máscara debe contener tres F T X'20's, o dos X'20 y uno X'21 '.

   j) Las comas se puede utilizar con la protección de cheques en la máscara. F T

   k) COBOL La documentación técnica de la máscara se muestra en este capítulo es similar en F T Cláusulas de consentimiento fundamentado previo.

l) La máscara de X'402021204B2020 'sería documentado como BZ99.99. F T

m) Corresponde a los programadores para determinar la ubicación de los puntos decimales desde F T toda la aritmética decimal empaquetado en BAL es la aritmética de enteros.

2. ¿Qué está mal con los siguientes editar máscaras?

   a) X'202020'

   b) X'402020'

   c) X'40202121'

   d) X'4020204B2020'

   e) X'4020204B2020206B2120'

   f) X'4020206B2020206B20212060'

3. Mostrar los cambios que haría a la rutina mostrada en el capítulo 8, a fin HDGS suprimir ceros a la izquierda en el recuento de la página.

4. Determinar la longitud correcta para el campo de la producción y el campo de empaquetado para cada uno de los siguientes editar las máscaras. A continuación mostramos el resultado si un campo de relleno de la longitud adecuada con un valor de -123 ha sido editado usando la máscara indicada.

```
Resultados de la salida de Bolsas de
a. X'40202120 '
b. X'4020206B202120 '
c. X'5C20206B2020214B2020 '
d. X'40202120C4C2 '
e. X'4020202020202020202060 '
f. X'5C20202020202120C3D9 '
g. X'40206B2020206B202120 '
h. X'402020204B2020C4C2 '
i. X'5C2020202020202060 '
j. X'40204B202020202020 '
```

5. Para cada una de las máscaras de la edición se muestra en el ejercicio 4, mostrará el resultado si un campo de relleno de la longitud adecuada y con un valor de cero fue editado. Repita para un campo repleto de una valor de -1.

6. Habida cuenta de las definiciones de los campos siguientes:

```
FLDA DC PL3'-12'
WK6  DS CL6
WK7  DS CL7
WK8  DS CL8
```

... Mostrar los resultados de las ediciones siguientes:

a) MVC WK6,=X'402020202020' ED WK6,FLDA
b) MVC WK6,=X'402020202120' ED WK6,FLDA
c) MVC WK6,=X'402021202020' ED WK6,FLDA
d) MVC WK6,=X'5C2020202120' ED WK6,FLDA
e) MVC WK6,=X'5C2021202020' ED WK6,FLDA
f) MVC WK7,=X'40202020202060' ED WK7,FLDA
g) MVC WK7,=X'5C202020212060' ED WK7,FLDA
h) MVC WK7,=X'4020206B202120' ED WK7,FLDA
i) MVC WK8,=X'4020206B20212060' ED WK8,FLDA
j) MVC WK8,=X'40206B2021204B20' ED WK8,FLDA
k) MVC WK8,=X'5C2021204B202060' ED WK8,FLDA
l) MVC WK8,=X'402021202020C3D9' ED WK8,FLDA

7. Escribir un programa que verificará sus respuestas al ejercicio 6. (Sugerencia: Véase en este capítulo.) EDITS9.MLC

8. Habida cuenta de las definiciones de los campos siguientes:

```
MASK DC X'4020206B2021204B202060'
WK11 DS CL11
```

Mostrar los resultados de las ediciones siguientes:

a) MVC WK11,MASK ED WK11,=PL4'0'
b) MVC WK11,MASK ED WK11,=PL4'1'
c) MVC WK11,MASK ED WK11,=PL4'-1'
d) MVC WK11,MASK ED WK11,=PL4'12'
e) MVC WK11,MASK ED WK11,=PL4'-12'
f) MVC WK11,MASK ED WK11,=PL4'1234'
g) MVC WK11,MASK ED WK11,=PL4'-1234'
h) MVC WK11,MASK ED WK11,=PL4'12345'
i) MVC WK11,MASK ED WK11,=PL4'123456'
j) MVC WK11,MASK ED WK11,=PL4'-1234567'

9. Escribir un programa que verificará sus respuestas al ejercicio 8 anterior. (Sugerencia: Véase en este capítulo.) EDITS9.MLC

10. (Este ejercicio es similar al ejercicio 12 del capítulo 7.) Modificar el programa COGS9A.MLC en este capítulo para incluir totales por estado, es decir, su resultado debería aparecer como sigue:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
                    COGSWORTH INDUSTRIES
                          Sales Recap
Product         Calif      Ill        Utah     Wisc     TOTAL
----------      -----     -----     -----     -----     -----
GIZMOS            20        30        20        20        90
WIDGETS           15        10        10         2        37
JUNQUE            25        15        15        18        73
----------      -----     -----     -----     -----     -----
TOTAL             60        55        45        40       200
      3 records processed.
```

11. Modifique el programa del ejercicio 13 del capítulo 7 de utilizar ED en lugar de UNPK y MVZ.

12. Modifique el programa del ejercicio 14 del capítulo 7 de utilizar ED en lugar de UNPK y MVZ.

13. (Consulte la base de datos pequeña ciudad en la nómina más conjuntos de datos.) Utilice el historial del archivo de producir un registro de nómina para el período de pago que termina 1/2/93. Las cifras totales incluyen y cuentan como se muestra. El informe debe aparecer como sigue:

```
.        1         2         3         4
12345678901234567890123456789012345678901234567890
            SMALL TOWN PAYROLL
     Payroll Register for PPED 1/2/93
     Employee    Hours         Gross
     --------   -------    -----------
       XXX      BZZ9.99    BZZ,ZZ9.99
       XXX      BZZ9.99    BZZ,ZZ9.99
       XXX      BZZ9.99    BZZ,ZZ9.99
     --------   -------    -----------
     TOTAL      BZZ9.99    BZZ,ZZ9.99
     There were BZZ9 checks printed.
```

Capítulo 10

Quiebre de la lógica de control de

Objetivos Al término de este capítulo usted será capaz de:

- Describir el escribir, rodar, cero, y ahorrar pasos utilizados en el procesamiento de control de descanso,
- Diseño y programación de un programa con un solo control de procesamiento a nivel de ruptura,
- Diseño y programación de un programa con múltiples niveles de control de procesamiento de descanso, y
- Diseño y programación de un programa mediante el control de descanso de procesamiento para producir un informe resumido.

Introducción En cada uno de nuestros programas hasta el momento ninguna cuenta y / o totales fueron mostrados al final del trabajo solamente. En este capítulo vamos a aprender a producir subtotales. Subtotales se puede hacer en cualquier número de niveles, y son el resultado del procesamiento de control de descanso. En este capítulo se presenta un solo nivel y múltiples niveles de control de procesamiento de descanso. No hay nuevas instrucciones se introducen en este capítulo.

A lo largo de este capítulo se hará uso del archivo de la oferta de cursos, OFFER.DAT. Nuestra primera de programa, OFFER10A.MLC, las listas de seleccionados los campos de cada registro en dicho archivo. La lógica de salto de página es incluido. Hay un límite de diez líneas de detalle en cada página. La salida se muestra a continuación, y el listado del programa completo se encuentra en la página siguiente. No hay nada nuevo aquí: el programa es dado el fin de proporcionar un punto de partida para nuestro debate.

```
          COURSE OFFERINGS      Page   1
   Sem   Course   Section   Teacher
   ---   ------   -------   -------
   W92    EG102      1        732
   W92    MA107      1        218
   W92    PE151      1        574
   F92    AC101      1        218
   F92    BU101      1        218
   F92    EG101      1        732
   F92    EG101      2        732
   F92    MA101      1        626
   F92    MA101      2        626
   F92    PE151      1        574
---------------------------------------
          COURSE OFFERINGS      Page   2
   Sem   Course   Section   Teacher
   ---   ------   -------   -------
   W93    EG102      1        854
   W93    MA101      1        626
   W93    MA107      1        626
   W93    PE151      1        574
```

222 ⟨*search..., read OFFER into IREC, EOFSW, SVREAD return* 222⟩≡ (223 232 241)

```
READ    EQU   *
        ST    R10,SVREAD
        GET   OFFER,IREC              Read a single offer record
        B     READX
ATEND   EQU   *
        MVI   EOFSW,C'Y'
READX   EQU   *
        L     R10,SVREAD
        BR    R10
```

223 ⟨*OFFER10A.MLC* 223⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: OFFER10A.MLC                                 *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce list of course offerings for all     *
*                    semesters. Includes page break logic.      *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'OFFER10A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'OFFER10A ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     OFFER+10,X'08'          PC/370 ONLY - Convert all
*                                       input from ASCII to EBCDIC
         OI     REPORT+10,X'08'         PC/370 ONLY - Convert all
*                                       output from EBCDIC to ASCII
         OPEN OFFER
         OPEN REPORT
         BAL    R10,READ
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         AP     PGS,=P'1'               Add 1 to page count
         MVC    HDPGS,=X'40202120' Edit pattern for page count
         ED     HDPGS,PGS               Move page count to heading
         PUT    REPORT,FORMFEED         PC/370 ONLY
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
         ZAP    LNS,=P'0'               Reset line count to zero
         L      R10,SVHDGS
```

```
                BR     R10
          ************************************************************
          *        PROCESS - Those things which happen once per record.  *
          ************************************************************
          PROCESS EQU     *
                  ST     R10,SVPROC
                  BAL    R10,CHKLNS
                  BAL    R10,FORMAT
                  BAL    R10,WRITE
                  BAL    R10,READ
                  L      R10,SVPROC
                  BR     R10
*************************************************************
*        READ - Read a record.                              *
*************************************************************
⟨search..., read OFFER into IREC, EOFSW, SVREAD return 222⟩
*************************************************************
*        CHKLNS - Check lines printed. Full page?           *
*************************************************************
⟨search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS 184⟩
          ************************************************************
          *        FORMAT - Format a single detail line.              *
          ************************************************************
          FORMAT   EQU   *
                   ST    R10,SVFORM
                   MVC   OREC(40),BLANKS
                   MVC   OSEM,ISEM                 Semester
                   MVC   OCID,ICID                 Course ID
                   MVC   OSECT,ISECT               Section number
                   MVC   OTID,ITID                 Teacher ID
                   MVC   OCRLF,WCRLF               PC/370 Only
                   L     R10,SVFORM
                   BR    R10
*************************************************************
*        WRITE - Write a single detail line.                *
*************************************************************
⟨search..., write REPORT from OREC, LNS++, SVWRITE return 185⟩
          ************************************************************
          *        WRAPUP - Those things which happen one time only,     *
          *                 after all records have been processed.      *
          ************************************************************
          WRAPUP   EQU   *
                   ST    R10,SVWRAP
                   CLOSE OFFER
                   CLOSE REPORT
                   WTO   'OFFER10A ... Course list on REPORT.TXT'
```

```
                L     R10,SVWRAP
                BR    R10
*****************************************************************
          *       Literals, if any, will go here                  *
          *****************************************************************
                LTORG
          *****************************************************************
          *       File definitions                                *
          *****************************************************************
          OFFER    DCB   LRECL=18,RECFM=F,MACRF=G,EODAD=ATEND,
                         DDNAME='OFFER.DAT'
          REPORT   DCB   LRECL=42,RECFM=F,MACRF=P,
                         DDNAME='REPORT.TXT'
*****************************************************************
*         RETURN ADDRESSES                                      *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
SVCHKLNS DC    F'0'                    CHKLNS
          *****************************************************************
          *       Miscellaneous field definitions                 *
          *****************************************************************
          WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
          EOFSW    DC    CL1'N'                   End of file? (Y/N)
          PGS      DC    PL2'0'                   Nbr of pages printed.
          LNS      DC    PL2'10'                  Lines printed on this page.
          MAXLNS   DC    PL2'10'                  Max nbr lines per page.
          *                                       My line counts exclude hdgs.
          BLANKS   DC    CL40' '
          *****************************************************************
          *       Input record definition                         *
          *****************************************************************
          IREC     DS    0CL18         1-18       Offer record
          ISEM     DS    CL3           1- 3       Semester
          ICID     DS    0CL5          4- 8       Course ID
          IDEPT    DS    CL2           4- 5       Department
                   DS    CL3           6- 8       Course number
          ISECT    DS    CL1           9- 9       Section number
          ITID     DS    CL3          10-12       Teacher ID
          IROOM    DS    CL4          13-16       Room number
          IOCRLF   DS    CL2          17-18       PC/370 only - CR/LF
          *****************************************************************
          *       Output (line) definition                        *
          *****************************************************************
          OREC     DS    0CL42         1-42
                   DC    CL3' '        1- 3
          OSEM     DS    CL3           4- 6       Semester
```

```
                DC    CL4' '        7-10
          OCID  DS    CL5          11-15         Course ID
                DC    CL6' '       16-21
          OSECT DS    CL1          22-22         Section number
                DC    CL8' '       23-30
          OTID  DS    CL3          31-33         Teacher ID
                DC    CL7' '       34-40
          OCRLF DS    CL2          41-42         PC/370 only - CR/LF
*************************************************************
       *         Headings definitions                          *
       *************************************************************
       *
       * --+--1--+--2--+--3--+--4
       *              COURSE OFFERINGS        PageBZZ9
       *
       *     Sem     Course    Section   Teachers
       *     --      ---    ----   ----
       *     XXX      XXXXX       X        XXX
       *     XXX      XXXXX       X        XXX
       *     XXX      XXXXX       X        XXX
       *
       FORMFEED DS      0CL42               PC/370 only
       *         DC     X'0C'               EBCDIC formfeed
       *         DC     CL39' '
                 DC     40C'_'              For testing...
                 DC     X'0D25'             EBCDIC CR/LF
       HD1       DS     0CL42
                 DC     CL36'           COURSE OFFERINGS       Page'
       HDPGS     DC     CL4'BZZ9'
                 DC     XL2'0D25'
       HD2       DS     0CL42
                 DC     CL40' '
                 DC     XL2'0D25'
       HD3       DS     0CL42
                 DC     CL40'    Sem   Course    Section   Teacher   '
                 DC     XL2'0D25'
       HD4       DS     0CL42
                 DC     CL40'    --   ---    ----   ----   '
                 DC     XL2'0D25'
                 END    BEGIN
```

10.1. Single Level Control Break - Problema de programación

Observe que el informe anterior es en la secuencia semestre. Vamos a modificar el informe de modo que sólo hay un semestre por página. Debido a que habrá sólo un semestre por página, sería redundante que incluyen el semestre en las líneas de detalle, así que vamos a dejar que la columna, y mostrar el semestre en el partidas en su lugar. Además, vamos a mostrar un recuento del número de secciones en un semestre en la final de la cada semestre. El nuevo informe aparecerá como sigue:

```
 ---------------------------------------
          COURSE OFFERINGS     Page   1
     Semester W92
        Course   Section      Teacher
          ------   -------      -------
           EG102      1            732
           MA107      1            218
           PE151      1            574
*** Sem W92    3 sections
 ---------------------------------------
          COURSE OFFERINGS      Page   2
          Semester F92
        Course   Section   Teacher
        ------   -------   -------
         AC101      1        218
         BU101      1        218
         EG101      1        732
         EG101      2        732
         MA101      1        626
         MA101      2        626
         PE151      1        574
    *** Sem F92    7 sections
---------------------------------------
          COURSE OFFERINGS      Page   3
            Semester W93
        Course   Section   Teacher
       ------   -------   -------
         EG102      1        854
         MA101      1        626
         MA107      1        626
         PE151      1        574
    *** Sem W93    4 sections
```

10.2. Single Level Control Break - Lógica

Los cambios necesarios para producir este informe es la siguiente:

1. Compruebe si hay un cambio en el semestre. Con el fin de ver si el semestre en un registro difiere de el semestre en el registro anterior, tendremos una variable en donde se puede "mantener" la semestre de registro a registro. Esto se define como:

```
HOLDSEM DC CL3' ' Hold semester
```

   Nos dijo que el semestre ahora sería una parte de las partidas:

```
HD2   DS 0CL42
      DC CL21'     Semester '
HDSEM DS CL3
      DC CL16' '
      DC XL2'0D25'
```

   ... así que antes de imprimir los títulos, nos trasladaremos HOLDSEM a HDSEM:

```
HDGS EQU *
     ST  R10,SVHDGS
     AP  PGS,=P'1'
     MVC HDPGS,=X'40202120'
     ED  HDPGS,PGS
     PUT REPORT,FORMFEED
     MVC HDSEM,HOLDSEM
     PUT REPORT,HD1
```

   Por supuesto, queremos que el semestre del primer registro para ser reconocidos como tales, de lo contrario vamos a tratar de imprimir los totales después de haber leído el primer registro único. Así, inicializamos HOLDSEM dentro de la rutina de instalación inmediatamente después de la lectura "cebado":

```
BAL R10,READ
MVC HOLDSEM,ISEM Control break
```

   Como se procesa cada registro, vamos a comprobar si hay cambios en el semestre. Dado que este es un cheque hecho para cada registro, que será, por supuesto, entran en la rutina de proceso (en contraposición a la SETUP o Wrapup.) Vamos a invocar una rutina por separado, CHKSEM, para ver si el semestre se ha cambiado. Lo hacemos sólo para mantener el proceso de rutina simple y ordenado:

```
PROCESS EQU *
        ST  R10,SVPROC
        BAL R10,CHKSEM See if new semester
```

... donde CHKSEM se define como:

```
*************************************************************
*        CHKSEM - Check for change in semester              *
*                 (control break)                           *
*************************************************************
```

229 ⟨*search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM, SVCHKSEM* 229⟩≡ (232)

```
CHKSEM   EQU   *
         ST    R10,SVCHKSEM
         CLC   HOLDSEM,ISEM              Compare w/ current
         BE    CHKSEMX                   Same semester, get out
         BAL   R10,ENDSEM                Process semester break
         MVC   HOLDSEM,ISEM              Update control break field
CHKSEMX EQU    *
         L     R10,SVCHKSEM
         BR    R10
```

De nuevo, esto se podría haber hecho dentro de la siguiente manera:

```
           ST      R10,SVPROC
           CLC     HOLDSEM,ISEM
           BE      PROCESS2
           BAL     R10,ENDSEM
           MVC     HOLDSEM,ISEM
PROCESS2 EQU       *
         (continued)
```

Tenga en cuenta que ha finalizado un semestre y comenzó otro, movemos el semestre en curso, ISEM, al campo de romper el control, HOLDSEM.

2. Si el semestre ha cambiado, extremo que el semestre y empezar otro. Vemos de CHKSEM que si el semestre ha cambiado, es decir, si no es igual a HOLDSEM ISEM, entonces invocamos romper la rutina de control, ENDSEM. Esta rutina de nuevo (a) imprimir el número de secciones de para este semestre, (b) restablecer las secciones contador a cero, y (c) la fuerza para el próximo semestre una nueva página:

```
ENDSEM EQU *
       ST  R10,SVENDSEM
       MVC OREC(40),BLANKS    This area used several ways
       BAL R10,WRITE          Skip a line
       MVC OREC+6(25),=C'*** Sem XXX BZZ9 sections'
```

```
MVC OREC+14(3),HOLDSEM
MVC OREC+18(4),=X'40202120'
ED  OREC+18(4),#SEM
BAL R10,WRITE
MVC OREC(40),BLANKS
ZAP #SEM,=P'0'          Reset counter
ZAP LNS,MAXLNS          Force next sem. to new page
L   R10,SVENDSEM
BR  R10
```

a) Imprimir el número de secciones para este semestre. Observe cómo se formatea el recuento. En primer lugar, la la línea está desactivada. (Esta es la misma superficie que se utilizó para los detalles, pero podemos volver a utilizarlo para que muestra la cuenta si nos gusta.) Por supuesto, cuando nos movemos espacios en blanco para OREC, lo hacemos de una longitud de 40 sólo para no destruir el CR / LF. De lo contrario, tendríamos que recordar para mover X'0D25 a OREC 40 (2). Se aleja una descripción de la cuenta. Observe cómo un etiqueta larga fue usado, con "XXX" indica que el semestre se van, y que indica BZZ9 cuando el recuento se irán. Que no es necesario hacerlo de esta manera. Me parece que esta técnica a ser muy útil, ya que es un poco auto-documentado. Esto es particularmente importante si vamos a utilizar el desplazamiento explícito y longitud como se muestra arriba. Sin embargo, la siguiente es funcionalmente equivalente y probablemente más comúnmente utilizadas:

```
MVC OREC+6(7),=C'*** Sem'
MVC OREC+14(3),HOLDSEM
MVC OREC+18(4),=X'40202120'
ED  OREC+18(4),#SEM
MVC OREC+23(8),=C'sections'
```

Después de haber utilizado la línea de detalle de esta manera, debemos recordar que limpiar después de nosotros mismos, que es decir, una vez más espacios en blanco para mover OREC (40).

Usted podría sentirse incómodo con la reutilización de la línea de detalle como lo hemos hecho aquí. Es muy comunes para definir un registro separado para este propósito:

```
MVC TOTSEM,HOLDSEM
MVC TOTCOUNT,=X'40202120'
ED  TOTCOUNT,#SEM
PUT REPORT,TOTREC
AP  LNS,=P'1'
```

donde ...

```
TOTREC   DS 0CL42
         DC CL6' '
```

```
            DC CL8'*** SEM '
TOTSEM      DC CL3' '
            DC CL1' '
TOTCOUNT    DC CL4' '
            DC CL18' sections'
            DC X'0D25'   CR/LF
```

Tenga en cuenta que usamos INFORME DE VENTA, TOTREC y el incremento de la línea de la lucha contra aquí en vez de BAL debido a que (escribir) de rutina pone OREC (sólo) para informar. R10, ESCRIBA 2

b) Restablecer el contador a cero secciones. Nos movemos a cero el contador para el número de secciones de dentro de este semestre, # SEM. De lo contrario, las secciones cuentan para W92 semestre se incluirá en las secciones en cuenta para el semestre de F93, y el recuento de las secciones para los semestres W92 y F93 serán incluidos en las secciones que cuentan para W93 semestre. 2

(c) de la Fuerza para el próximo semestre una nueva página. Obligar a cada nuevo semestre a una nueva página es sencillo: basta con fijar la línea de la lucha contra el máximo permitido. Recuerde que antes de imprimir cualquier líneas de detalle, siempre invocar la CHKLNS de rutina para ver si una página es completa y, de ser así, la impresión partidas. Al establecer la línea contraria a la máxima permitida, la página será considerado completo y de las partidas serán impresos, con el nuevo semestre, antes de imprimir el siguiente detalle línea.

3. Como se procesa cada registro, agregar uno al número de secciones para ese semestre. Esta es la hecho para cada registro y, como tal, se incluye en la rutina PROCESO. Se lleva a cabo después de la verificación de un nuevo semestre, de lo contrario el semestre equivocada puede obtener "crédito" para este registro.

```
PROCESS EQU *
        ST  R10,SVPROC
        BAL R10,CHKSEM See if new semester
        AP  #SEM,=P'1' Count sections by semester
```

4. Después de todos los registros han sido procesados, imprimir una última serie de los totales. Dentro de la Wrapup invocamos a la rutina de romper el control de rutina por última vez: no porque un nuevo semestre ha comenzado, sino porque el último semestre ha terminado. Si no lo hacemos así, entonces no vamos a obtener una recuento de las secciones para el último semestre.

```
WRAPUP EQU   *
       ST    R10,SVWRAP
```

```
         BAL   R10,ENDSEM
         CLOSE OFFER
         CLOSE REPORT
```

pero debe hacerse antes del cierre Esto se puede hacer antes o después del cierre de la OFERTA el archivo de informe.

10.3. Single Level Control Break - Program Solution

232 ⟨*OFFER10B.mlc* 232⟩≡

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: OFFER10B.MLC                                 *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce list of course offerings for all     *
*                  semesters. Includes page break logic.        *
*                  Single level control break example.          *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'OFFER10B ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'OFFER10B ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     OFFER+10,X'08'          PC/370 ONLY - Convert all
*                                       input from ASCII to EBCDIC
         OI     REPORT+10,X'08'         PC/370 ONLY - Convert all
*                                       output from EBCDIC to ASCII
         OPEN OFFER
         OPEN REPORT
         BAL    R10,READ
         MVC    HOLDSEM,ISEM            Control break
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                 *
```

```
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         AP     PGS,=P'1'                Add 1 to page count
         MVC    HDPGS,=X'40202120' Edit pattern for page count
         ED     HDPGS,PGS                Move page count to heading
         PUT    REPORT,FORMFEED          PC/370 ONLY
         MVC    HDSEM,HOLDSEM
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
         PUT    REPORT,HD5
         ZAP    LNS,=P'0'                Reset line count to zero
         L      R10,SVHDGS
         BR     R10
*****************************************************************
*        PROCESS - Those things which happen once per record.  *
*****************************************************************
PROCESS EQU    *
         ST     R10,SVPROC
         BAL    R10,CHKSEM               See if new semester
         AP     #SEM,=P'1'               Count sections by semester
         BAL    R10,CHKLNS
         BAL    R10,FORMAT
         BAL    R10,WRITE
         BAL    R10,READ
         L      R10,SVPROC
         BR     R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
⟨search..., read OFFER into IREC, EOFSW, SVREAD return 222⟩
*****************************************************************
*        CHKSEM - Check for change in semester                  *
*                 (control break)                               *
*****************************************************************
⟨search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM, SVCHKSEM 229⟩
         *****************************************************************
         *        ENDSEM - End semester                                  *
         *                 (Process control break)                       *
         *                 Show count of sections for this semester.     *
         *                 Force next semester to another page.          *
         *****************************************************************
         ENDSEM   EQU    *
                  ST     R10,SVENDSEM
```

```
                MVC   OREC(40),BLANKS          This area used several ways
                BAL   R10,WRITE                Skip a line
                MVC   OREC+6(25),=C'*** Sem XXX BZZ9 sections'
                MVC   OREC+14(3),HOLDSEM
                MVC   OREC+18(4),=X'40202120'
                ED    OREC+18(4),#SEM
                BAL   R10,WRITE
                MVC   OREC(40),BLANKS
                ZAP   #SEM,=P'0'               Reset counter
                ZAP   LNS,MAXLNS               Force next sem. to new page
                L     R10,SVENDSEM
                BR    R10
*************************************************************
*        CHKLNS - Check lines printed. Full page?           *
*************************************************************
```

⟨*search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS* 184⟩

```
         *************************************************************
         *        FORMAT - Format a single detail line.              *
         *************************************************************
         FORMAT  EQU   *
                 ST    R10,SVFORM
                                                Removed MVC OSEM,ISEM
                 MVC   OREC(40),BLANKS
                 MVC   OCID,ICID               Course ID
                 MVC   OSECT,ISECT             Section number
                 MVC   OTID,ITID               Teacher ID
                 MVC   OCRLF,WCRLF             PC/370 Only
                 L     R10,SVFORM
                 BR    R10
         *************************************************************
         *        WRITE - Write a single detail line.                *
         *************************************************************
```

⟨*search..., write REPORT from OREC, LNS++, SVWRITE return* 185⟩

```
         *************************************************************
         *        WRAPUP - Those things which happen one time only,  *
         *                 after all records have been processed.    *
         *************************************************************
         WRAPUP  EQU   *
                 ST    R10,SVWRAP
                 BAL   R10,ENDSEM              Final control break process
                 CLOSE OFFER
                 CLOSE REPORT
                 WTO   'OFFER10B ... Course list on REPORT.TXT'
                 L     R10,SVWRAP
                 BR    R10
         *************************************************************
```

```
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
OFFER    DCB   LRECL=18,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='OFFER.DAT'
REPORT   DCB   LRECL=42,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
SVCHKLNS DC    F'0'                    CHKLNS
SVCHKSEM DC    F'0'                    CHKSEM
SVENDSEM DC    F'0'                    ENDSEM
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                 PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                  End of file? (Y/N)
PGS      DC    PL2'0'                  Nbr of pages printed.
LNS      DC    PL2'10'                 Lines printed on this page.
MAXLNS   DC    PL2'10'                 Max nbr lines per page.
*                                      My line counts exclude hdgs.
BLANKS   DC    CL40' '
HOLDSEM DC     CL3' '                  Hold semester
#SEM     DC    PL2'0'                  Sections in a semester
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL18          1-18     Offer record
ISEM     DS    CL3            1- 3     Semester
ICID     DS    0CL5           4- 8     Course ID
IDEPT    DS    CL2            4- 5     Department
         DS    CL3            6- 8     Course number
ISECT    DS    CL1            9- 9     Section number
ITID     DS    CL3           10-12     Teacher ID
IROOM    DS    CL4           13-16     Room number
IOCRLF   DS    CL2           17-18     PC/370 only - CR/LF
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL42          1-42
         DC    CL10' '        1-10
```

```
OCID     DS    CL5          11-15         Course ID
         DC    CL6' '       16-21
OSECT    DS    CL1          22-22         Section number
         DC    CL8' '       23-30
OTID     DS    CL3          31-33         Teacher ID
         DC    CL7' '       34-40
OCRLF    DS    CL2          41-42         PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
*
* --+--1--+--2--+--3--+--4
*            COURSE OFFERINGS              PageBZZ9
*              Semester XXX
*
*           Course    Section        Teachers
*           ---    ----        ----
*            XXXXX        X             XXX
*            XXXXX        X             XXX
*            XXXXX        X             XXX
*
*            Sem XXX BZZ9 sections
*
FORMFEED DS   0CL42
*        DC   X'0C'
*        DC   CL39' '
         DC   40C'_'
         DC   X'0D25'
HD1      DS   0CL42
         DC   CL36'
HDPGS    DC   CL4'BZZ9'
         DC   XL2'0D25'
HD2      DS   0CL42
         DC   CL21'
HDSEM    DS   CL3
         DC   CL16' '
         DC   XL2'0D25'
HD3      DS   0CL42
         DC   CL40' '
         DC   XL2'0D25'
HD4      DS   0CL42
         DC   CL40'
         DC   XL2'0D25'
HD5      DS   0CL42
         DC   CL40'
         DC   XL2'0D25'
```

```
END  BEGIN
```

10.4. Romper el control de nivel múltiple - Problema de programación

Ahora introducimos múltiples niveles de control de procesamiento de descanso. Supongamos que, además de mostrar un recuento de las secciones por semestre, también nos gustaría mostrar un recuento de las secciones por departamento dentro de semestre, por ejemplo, después de enumerar todas las secciones que ofrece la Matemáticas departamento, se mostrará un recuento de las secciones. Nosotros, sin embargo, muestran esta cuenta sólo si el contador es mayor que 1, a fin de no el desorden de nuestro informe. Además, siempre se duplicará el espacio (saltar una línea) entre los departamentos. El nuevo informe aparecerá como se muestra en la siguiente página.

10.4.1. Romper el control de nivel múltiple - Lógica

La lógica necesaria para producir este informe es similar a la utilizada en la simple ruptura nivel de control transformación y es el siguiente:

1. Compruebe si hay un cambio en el departamento. Así como hemos solicitado un cambio en el semestre en el el programa anterior, tenemos que comprobar si hay cambios en el departamento en este programa. Con el fin de Para ello, tendremos que una variable en donde se puede "mantener" el departamento de registro a registro. Esto se define como:

   ```
   HOLDDEPT DC CL2' ' Hold department
   ```

   Por supuesto, queremos que el semestre y el departamento del primer registro para ser reconocidos como tales, por lo que inicializar HOLDSEM y HOLDDEPT dentro de la rutina de instalación después de la imprimación léase:

10.4.2. La producción de OFFER10C.MLC - Ejemplo de nivel múltiple romper el control de

```
          COURSE OFFERINGS      Page   1
            Semester W92
         Course   Section   Teacher
         ------   -------   -------
          EG102      1        732
          MA107      1        218
          PE151      1        574
       *** Sem W92    3 sections
-------------------------------------
```

```
            COURSE OFFERINGS      Page   2
              Semester F92
           Course   Section   Teacher
           ------   -------   -------
            AC101      1        218
            BU101      1        218
            EG101      1        732
            EG101      2        732
         ** Dept EG    2 sections
            MA101      1        626
----------------------------------------
            COURSE OFFERINGS      Page   3
              Semester F92
           Course   Section   Teacher
           ------   -------   -------
            MA101      2        626
         ** Dept MA    2 sections
            PE151      1        574
        *** Sem F92    7 sections
----------------------------------------
            COURSE OFFERINGS      Page   4
              Semester W93
           Course   Section  Teacher
           ------   -------   -------
            EG102      1        854
            MA101      1        626
            MA107      1        626
         ** Dept MA    2 sections
            PE151      1        574
        *** Sem W93    4 sections
BAL R10,READ
MVC HOLDSEM,ISEM   Major control break
MVC HOLDDEPT,IDEPT Minor control break
```

Como se procesa cada registro, vamos a comprobar si hay cambios en el semestre o el departamento. Desde que este es un hecho de verificación para cada registro, que estará dentro de la rutina PROCESO. Vamos a invocar a CHKSEM para ver si el semestre ha cambiado, e invocar CHKDEPT para ver si el departamento ha cambiado.

Es importante señalar que un cambio en el semestre implica un cambio en el departamento. Para ejemplo, si el departamento que figuran en el último semestre fue W92 PE, y el primer departamento que figuran en el semestre fue también F93 PE, estos aparecerían en dos páginas distintas y tener dos cuentas separadas. Por lo tanto, comprobar el semestre antes de revisar el departamento:

```
PROCESS EQU *
        ST  R10,SVPROC
        BAL R10,CHKSEM  See if new semester
        BAL R10,CHKDEPT See if new department
```

donde CHKSEM y CHKDEPT son ...

```
*****************************************************************
*        CHKSEM - Check for change in semester                  *
*                 (major control break)                         *
*****************************************************************
```

239a ⟨*search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM+IDEPT, SVCHKSEM* 239a⟩≡ (241 249)

```
CHKSEM   EQU   *
         ST    R10,SVCHKSEM
         CLC   HOLDSEM,ISEM              Compare w/ current
         BE    CHKSEMX                   Same semester, get out
         BAL   R10,ENDSEM                Process semester break
         MVC   HOLDSEM,ISEM              Update major break field
         MVC   HOLDDEPT,IDEPT            Update minor break field
CHKSEMX  EQU    *
         L     R10,SVCHKSEM
         BR    R10
```

```
*****************************************************************
*        CHKDEPT - Check for change in department               *
*                  (minor control break)                        *
*****************************************************************
```

239b ⟨*search..., CHKDEPT, HOLDDEPT vs IDEPT to ENDDEPT,SVCHKDEP* 239b⟩≡ (241 249)

```
CHKDEPT EQU    *
                    ST    R10,SVCHKDEP
                    CLC   HOLDDEPT,IDEPT            Compare w/ current
                    BE    CHKDEPTX                  Same department, get out
                    BAL   R10,ENDDEPT               Process department break
                    MVC   HOLDDEPT,IDEPT            Update control break field
          CHKDEPTX EQU    *
                    L     R10,SVCHKDEP
                    BR    R10
```

En múltiples niveles de control de procesamiento de descanso, siempre evaluar el control campos de ruptura de mayor a menor!

2. Si el semestre ha cambiado, extremo que el semestre y empezar otro. El tratamiento aquí es el mismo que en el programa anterior, salvo un

cambio en el semestre implica un cambio en el departamento. Así que lo primero que hacemos dentro de ENDSEM es poner fin a la anterior y el departamento comenzar el siguiente departamento:

```
ENDSEM EQU *
       ST  R10,SVENDSEM
       BAL R10,ENDDEPT        Change in semester implies
*                             change in department as well.
       MVC OREC(40),BLANKS    This area used several ways
       MVC OREC+6(25),=C'*** Sem XXX BZZ9 sections'
```

(ENDDEPT se discute a continuación.)

En múltiples niveles de control de procesamiento de descanso, una ruptura en un nivel implica una ruptura en todos los niveles más bajos!

3. Si el Departamento ha cambiado, entonces fin de que los departamentos y comenzar otra. Vemos a partir de que si el Departamento ha cambiado, es decir, si HOLDDEPT no es igual a IDEPT, a continuación, CHKDEPT invocamos a romper la rutina de control, ENDDEPT. Esta rutina de nuevo (a) imprimir el número de de las secciones de este departamento (si dos o más de acuerdo con las especificaciones dadas arriba), (b) imprimir una línea en blanco (sin importar el número de secciones de este departamento), y (c) restablecer el departamento de secciones de nivel el contador a cero:

```
*************************************************************
*        ENDDEPT- End department                            *
*                 (Process minor control break)             *
*                 Print count of sections in department     *
*                 if that count is two or more.             *
*                 Regardless, skip a line.                  *
*************************************************************
ENDDEPT EQU    *
         ST    R10,SVENDDEP
         CP    #DEPT,=P'2'
         BL    ENDDEPT2
         MVC   OREC(40),BLANKS    This area used several ways
         BAL   R10,WRITE          Blank line before count
         MVC   OREC+7(24),=C'** Dept XX BZZ9 sections'
         MVC   OREC+15(2),HOLDDEPT
         MVC   OREC+18(4),=X'40202120'
         ED    OREC+18(4),#DEPT
         BAL   R10,WRITE
ENDDEPT2 EQU   *
         MVC   OREC(40),BLANKS
```

```
BAL   R10,WRITE
ZAP   #DEPT,=P'0'        Reset counter
L     R10,SVENDDEP
BR    R10
```

Si queremos un conteo por departamento, o no lo que cuenta es mayor que uno, entonces sólo tenemos que quitar la CP y las instrucciones de BL. (Esta sección no se examinarán en detalle que el razonamiento es similar a la utilizada en la discusión de ENDSEM en los últimos programa.)

4. Como se procesa cada registro, agregar uno al número de secciones para que el semestre y departamento. Esto se hace para cada registro y, como tal, se incluye en la rutina PROCESO. Se lleva a cabo después de la verificación de un nuevo semestre o el nuevo departamento.

```
PROCESO EQU *
        ST  R10,SVPROC
        BAL R10,CHKSEM  See if new semester
        BAL R10,CHKDEPT See if new department
        AP  #SEM,=P'1'  Count sections by semester
        AP  #DEPT,=P'1' Count sections by department
```

5. Después de todos los registros han sido procesados, imprimir una última serie de los totales. Dentro de la Wrapup rutina que invocamos ENDSEM una última vez, al igual que en el programa anterior. Tenga en cuenta que lo que hacemos No es necesario invocar ENDDEPT, ya que ENDSEM lo hará. Por lo tanto, tendremos los totales de la Departamento de última (si hay más de 1 sección se ofrece), así como los totales para el último semestre. La rutina es la misma que en el programa anterior:

```
WRAPUP EQU   *
       ST    R10,SVWRAP
       BAL   R10,ENDSEM
       CLOSE OFFER
       CLOSE REPORT
```

10.4.3. Romper el control de nivel múltiple - Programa de Solución

241 ⟨*OFFER10C.mlc* 241⟩≡

```
        PRINT NOGEN
*****************************************************************
*       FILENAME: OFFER10C.MLC                                  *
*       AUTHOR : Bill Qualls                                    *
*       SYSTEM : PC/370 R4.2                                    *
*       REMARKS : Produce list of course offerings for all      *
*                 semesters. Includes page break logic.         *
```

```
*                    Multiple level control break example.     *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'OFFER10C ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'OFFER10C ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     OFFER+10,X'08'            PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN OFFER
         OPEN REPORT
         BAL    R10,READ
         MVC    HOLDSEM,ISEM              Major control break
         MVC    HOLDDEPT,IDEPT            Minor control break
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         AP     PGS,=P'1'                 Add 1 to page count
         MVC    HDPGS,=X'40202120' Edit pattern for page count
         ED     HDPGS,PGS                 Move page count to heading
         PUT    REPORT,FORMFEED           PC/370 ONLY
         MVC    HDSEM,HOLDSEM
         PUT    REPORT,HD1
         PUT    REPORT,HD2
         PUT    REPORT,HD3
         PUT    REPORT,HD4
         PUT    REPORT,HD5
         ZAP    LNS,=P'0'                 Reset line count to zero
         L      R10,SVHDGS
         BR     R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
```

```
*****************************************************************
PROCESS EQU     *
                ST      R10,SVPROC
                BAL     R10,CHKSEM              See if new semester
                BAL     R10,CHKDEPT             See if new department
                AP      #SEM,=P'1'              Count sections by semester
                AP      #DEPT,=P'1'             Count sections by department
                BAL     R10,CHKLNS
                BAL     R10,FORMAT
                BAL     R10,WRITE
                BAL     R10,READ
                L       R10,SVPROC
                BR      R10
*****************************************************************
*       READ - Read a record.                                   *
*****************************************************************
⟨search..., read OFFER into IREC, EOFSW, SVREAD return 222⟩
*****************************************************************
*       CHKSEM - Check for change in semester                   *
*                (major control break)                          *
*****************************************************************
⟨search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM+IDEPT, SVCHKSEM 239a⟩
*****************************************************************
*       CHKDEPT - Check for change in department                *
*                 (minor control break)                         *
*****************************************************************
⟨search..., CHKDEPT, HOLDDEPT vs IDEPT to ENDDEPT,SVCHKDEP 239b⟩
                *************************************************************
                *       ENDSEM - End semester                                 *
                *                (Process major control break)                *
                *                Show count of sections for this semester.    *
                *                Force next semester to another page.         *
                *************************************************************
                ENDSEM  EQU     *
                        ST      R10,SVENDSEM
                        BAL     R10,ENDDEPT             Change in semester implies
                *                                       change in department as well.
                        MVC     OREC(40),BLANKS         This area used several ways
                        MVC     OREC+6(25),=C'*** Sem XXX BZZ9 sections'
                        MVC     OREC+14(3),HOLDSEM
                        MVC     OREC+18(4),=X'40202120'
                        ED      OREC+18(4),#SEM
                        BAL     R10,WRITE
                        MVC     OREC(40),BLANKS
                        ZAP     #SEM,=P'0'              Reset counter
                        ZAP     LNS,MAXLNS              Force next sem. to new page
```

```
                L     R10,SVENDSEM
                BR    R10
          ***************************************************************
          *        ENDDEPT- End department                                *
          *                 (Process minor control break)                 *
          *                 Print count of sections in department         *
          *                 if that count is two or more.                 *
          *                 Regardless, skip a line.                      *
          ***************************************************************
          ENDDEPT  EQU   *
                   ST    R10,SVENDDEP
                   CP    #DEPT,=P'2'
                   BL    ENDDEPT2
                   MVC   OREC(40),BLANKS           This area used several ways
                   BAL   R10,WRITE                 Blank line before count
                   MVC   OREC+7(24),=C'** Dept XX BZZ9 sections'
                   MVC   OREC+15(2),HOLDDEPT
                   MVC   OREC+18(4),=X'40202120'
                   ED    OREC+18(4),#DEPT
                   BAL   R10,WRITE
          ENDDEPT2 EQU   *
                   MVC   OREC(40),BLANKS
                   BAL   R10,WRITE
                   ZAP   #DEPT,=P'0'               Reset counter
                   L     R10,SVENDDEP
                   BR    R10
*************************************************************
*        CHKLNS - Check lines printed. Full page?           *
*************************************************************
⟨search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS 184⟩
          ***************************************************************
          *        FORMAT - Format a single detail line.                  *
          ***************************************************************
          FORMAT   EQU   *
                   ST    R10,SVFORM
                   MVC   OREC(40),BLANKS
                   MVC   OCID,ICID                 Course ID
                   MVC   OSECT,ISECT               Section number
                   MVC   OTID,ITID                 Teacher ID
                   MVC   OCRLF,WCRLF               PC/370 Only
                   L     R10,SVFORM
                   BR    R10
*************************************************************
*        WRITE - Write a single detail line.                *
*************************************************************
⟨search..., write REPORT from OREC, LNS++, SVWRITE return 185⟩
```

```
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         BAL   R10,ENDSEM                Final control break process
         CLOSE OFFER
         CLOSE REPORT
         WTO   'OFFER10C ... Course list on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
OFFER    DCB   LRECL=18,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='OFFER.DAT'
REPORT   DCB   LRECL=42,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
SVCHKLNS DC    F'0'                    CHKLNS
SVCHKSEM DC    F'0'                    CHKSEM
SVCHKDEP DC    F'0'                    CHKDEPT
SVENDSEM DC    F'0'                    ENDSEM
SVENDDEP DC    F'0'                    ENDDEPT
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                 PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                  End of file? (Y/N)
PGS      DC    PL2'0'                  Nbr of pages printed.
LNS      DC    PL2'10'                 Lines printed on this page.
MAXLNS   DC    PL2'10'                 Max nbr lines per page.
*                                      My line counts exclude hdgs.
BLANKS   DC    CL40' '
HOLDSEM DC     CL3' '                  Hold semester
HOLDDEPT DC    CL2' '                  Hold department
#SEM     DC    PL2'0'                  Sections in a semester
#DEPT    DC    PL2'0'                  Sections in a semester/dept
```

```
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL18         1-18         Offer record
ISEM     DS    CL3           1- 3         Semester
ICID     DS    0CL5          4- 8         Course ID
IDEPT    DS    CL2           4- 5         Department
         DS    CL3           6- 8         Course number
ISECT    DS    CL1           9- 9         Section number
ITID     DS    CL3          10-12         Teacher ID
IROOM    DS    CL4          13-16         Room number
IOCRLF   DS    CL2          17-18         PC/370 only - CR/LF
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL42         1-42
         DC    CL10' '       1-10
OCID     DS    CL5          11-15         Course ID
         DC    CL6' '       16-21
OSECT    DS    CL1          22-22     Section number
         DC    CL8' '       23-30
OTID     DS    CL3          31-33     Teacher ID
         DC    CL7' '       34-40
OCRLF    DS    CL2          41-42     PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
*
* --+--1--+--2--+--3--+--4
*             COURSE OFFERINGS         PageBZZ9
*               Semester XXX
*
*            Course    Section     Teachers
*            ---    ----     ----
*             XXXXX       X          XXX
*
*             XXXXX       X          XXX
*             XXXXX       X          XXX
*
*             Dept XX BZZ9 sections
*
*             Sem XXX BZZ9 sections
*
FORMFEED DS     0CL42                PC/370 only
*        DC     X'0C'                EBCDIC formfeed
*        DC     CL39' '
```

```
            DC     40C'_'                For testing...
            DC     X'0D25'               EBCDIC CR/LF
HD1         DS     0CL42
            DC     CL36'              COURSE OFFERINGS       Page'
HDPGS       DC     CL4'BZZ9'
            DC     XL2'0D25'
HD2         DS     0CL42
            DC     CL21'                Semester '
HDSEM       DS     CL3
            DC     CL16' '
            DC     XL2'0D25'
HD3         DS     0CL42
            DC     CL40' '
            DC     XL2'0D25'
HD4         DS     0CL42
            DC     CL40'           Course    Section    Teacher   '
            DC     XL2'0D25'
HD5         DS     0CL42
            DC     CL40'           ---    ----    ----   '
            DC     XL2'0D25'
            END    BEGIN
```

10.4.4. Totales Rolling

No es más que una manera de mantener el recuento de las secciones por departamento y por semestre. En nuestro solución hemos añadido una a cada uno de estos dos contadores en la rutina PROCESO. También es común a incrementar el contador de poca importancia (el contador es de menor importancia, en este caso, el recuento de las secciones de departamento, # DEPT). Luego, cuando se detecta un periodo de control de menor importancia, agregue la cuenta de menor importancia a la contar con mayores, antes de restablecer el recuento de menores a cero. Esto se refiere a veces como "renovable" los totales. La aplicación de esta "lógica de trabajo" sería la siguiente:

```
PROCESS EQU *
        ST  R10,SVPROC
        BAL R10,CHKSEM                        Have removed AP #SEM,=P'1'
        BAL R10,CHKDEPT
        AP  #DEPT,=P'1'
```

y ...

```
ENDDEPT EQU *
        .
        .
        .
```

```
BAL R10,WRITE
AP  #SEM,#DEPT   Roll counts
ZAP #DEPT,=P'0'  Reset counter
L   R10,SVENDDEP
BR  R10
```

Esta técnica es realmente más eficiente que el método anterior. Hay un total de dos puntos de acceso, por lo que el tamaño del programa es esta misma. Pero en el viejo método, # SEM se incrementa (por uno) de una vez por cada registro. En este método, # SEM se incrementa (por la cantidad en # DEPT) una vez por cada departamento. La técnica a utilizar es una cuestión de preferencia personal, pero debe sin duda ser capaz de reconocer y entender tanto.

10.5. Informe resumido - Problema de programación

En nuestra versión final del programa que desea producir un informe resumido sólo, es decir, en lugar de lista de cada curso y sección ofrece durante un semestre dado, vamos a mostrar cada departamento y el número de secciones ofrecidas por ese departamento. El nuevo informe aparecerá como sigue:

```
COURSE OFFERINGS            Page        1
                    Semester W92
                  Dept     Sections
                 ------    --------
                  EG           1
                  MA           1
                  PE           1
                 ------    --------
                  Total        3
          ----------------------------------------
                  COURSE OFFERINGS            Page        2
                     Semester F92
                  Dept     Sections
                 ------    --------
                  AC           1
                  BU           1
                  EG           2
                  MA           2
                  PE           1
                 ------    --------
                  Total        7
        ----------------------------------------
                COURSE OFFERINGS      Page   3
```

```
   Semester W93
 Dept     Sections
------    --------
 EG           1
 MA           2
 PE           1
------    --------
 Total        4
```

10.5.1. Informe resumido - Lógica

Aunque en este informe difiere considerablemente del informe anterior, los cambios necesarios para elaborar este informe son realmente trivial. En primer lugar, no hay informes de nivel de registro, por lo que la de ramificación y enlace con el formato y las rutinas ESCRIBA han quitado de la rutina PROCESO. En De hecho, la rutina de formato se ha retirado del programa por completo. En segundo lugar, el formato de la cuenta se ha cambiado en el ENDSEM y rutinas ENDDEPT. Todos los demás cambios son para el formato de solamente.

10.5.2. Informe resumido - Solución de programa

249 ⟨*ff* 40⟩+≡ ◁207 289▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: OFFER10D.MLC                                 *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce list of course offerings for all     *
*                  semesters. Includes page break logic.        *
*                  Multiple level control break.                *
*                  Minor break (dept) is summary only.          *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'OFFER10D ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'OFFER10D ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                before any records are processed.              *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     OFFER+10,X'08'          PC/370 ONLY - Convert all
```

```
*                                       input from ASCII to EBCDIC
         OI    REPORT+10,X'08'          PC/370 ONLY - Convert all
*                                       output from EBCDIC to ASCII
         OPEN OFFER
         OPEN REPORT
         BAL   R10,READ
         MVC   HOLDSEM,ISEM             Major control break
         MVC   HOLDDEPT,IDEPT           Minor control break
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU   *
         ST    R10,SVHDGS
         AP    PGS,=P'1'                Add 1 to page count
         MVC   HDPGS,=X'40202120' Edit pattern for page count
         ED    HDPGS,PGS                Move page count to heading
         PUT   REPORT,FORMFEED          PC/370 ONLY
         MVC   HDSEM,HOLDSEM
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         PUT   REPORT,HD5
         ZAP   LNS,=P'0'                Reset line count to zero
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.  *
*****************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         BAL   R10,CHKSEM               See if new semester
         BAL   R10,CHKDEPT              See if new department
         AP    #SEM,=P'1'               Count be semester
         AP    #DEPT,=P'1'              Count by semester/dept
                                         No detail-level reporting
         BAL   R10,READ
                                         Removed BAL to FORMAT and WRITE
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
READ     EQU   *
```

```
                ST    R10,SVREAD
                GET   OFFER,IREC                Read a single offer record
                B     READX
          ATEND EQU   *
                MVI   EOFSW,C'Y'
          READX EQU   *
                L     R10,SVREAD
                BR    R10
*************************************************************
*       CHKSEM - Check for change in semester               *
*                (major control break)                      *
*************************************************************
```

⟨*search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM+IDEPT, SVCHKSEM* 239a⟩

```
*************************************************************
*       CHKDEPT - Check for change in department            *
*                 (minor control break)                     *
*************************************************************
```

⟨*search..., CHKDEPT, HOLDDEPT vs IDEPT to ENDDEPT,SVCHKDEP* 239b⟩

```
          *************************************************************
          *       ENDSEM - End semester                               *
          *                (Process major control break)              *
          *                Show count of sections for this semester.  *
          *                Force next semester to another page.       *
          *************************************************************
          ENDSEM  EQU   *
                  ST    R10,SVENDSEM
                  BAL   R10,ENDDEPT               Change in semester implies
          *                                       change in department as well.
                  PUT   REPORT,HD5
                  AP    LNS,=P'1'
                  MVC   OREC(40),BLANKS           Reuse this line
                  MVC   ODEPT(5),=CL5'Total'
                  MVC   O#DEPT,=X'40202120'
                  ED    O#DEPT,#SEM
                  BAL   R10,WRITE
                  MVC   OREC(40),BLANKS
                  ZAP   #SEM,=P'0'                Reset semester record count
                  ZAP   LNS,MAXLNS                Force next sem. to new page
                  L     R10,SVENDSEM
                  BR    R10
          *************************************************************
          *       ENDDEPT- End department                             *
          *                (Process minor control break)              *
          *                Print a count of courses (sections) in     *
          *                this department.                           *
          *************************************************************
```

```
ENDDEPT EQU    *
        ST     R10,SVENDDEP
        BAL    R10,CHKLNS
        MVC    OREC(40),BLANKS
        MVC    ODEPT,HOLDDEPT          Department
        MVC    O#DEPT,=X'40202120'
        ED     O#DEPT,#DEPT            How many this department?
        MVC    OCRLF,WCRLF             PC/370 only
        BAL    R10,WRITE
        ZAP    #DEPT,=P'0'             Reset dept record count
        L      R10,SVENDDEP
        BR     R10
*************************************************************
*       CHKLNS - Check lines printed. Full page?            *
*************************************************************
```

⟨*search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS* 184⟩

```
        *************************************************************
        *       WRITE - Write a single detail line.                 *
        *************************************************************
        WRITE   EQU    *
                ST     R10,SVWRITE
                PUT    REPORT,OREC             Write report line
                AP     LNS,=P'1'
                L      R10,SVWRITE
                BR     R10
        *************************************************************
        *       WRAPUP - Those things which happen one time only,   *
        *                  after all records have been processed.   *
        *************************************************************
        WRAPUP  EQU    *
                ST     R10,SVWRAP
                BAL    R10,ENDSEM              Final control break process
                CLOSE  OFFER
                CLOSE  REPORT
                WTO    'OFFER10D ... Course list on REPORT.TXT'
                L      R10,SVWRAP
                BR     R10
        *************************************************************
        *       Literals, if any, will go here                      *
        *************************************************************
                LTORG
        *************************************************************
        *       File definitions                                                *
        *************************************************************
        OFFER   DCB    LRECL=18,RECFM=F,MACRF=G,EODAD=ATEND,
                       DDNAME='OFFER.DAT'
```

```
          REPORT   DCB    LRECL=42,RECFM=F,MACRF=P,
                          DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
SVCHKLNS DC    F'0'                    CHKLNS
SVCHKSEM DC    F'0'                    CHKSEM
SVCHKDEP DC    F'0'                    CHKDEPT
SVENDSEM DC    F'0'                    ENDSEM
SVENDDEP DC    F'0'                    ENDDEPT
          *****************************************************************
          *        Miscellaneous field definitions                        *
          *****************************************************************
          WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
          EOFSW    DC    CL1'N'                   End of file? (Y/N)
          PGS      DC    PL2'0'                   Nbr of pages printed.
          LNS      DC    PL2'10'                  Lines printed on this page.
          MAXLNS   DC    PL2'10'                  Max nbr lines per page.
          *                                       My line counts exclude hdgs.
          BLANKS   DC    CL40' '
          HOLDSEM DC     CL3' '                   Hold semester
          HOLDDEPT DC    CL2' '                   Hold department
          #SEM     DC    PL2'0'                   Sections in a semester
          #DEPT    DC    PL2'0'                   Sections in a semester/dept
          *****************************************************************
          *        Input record definition                                *
          *****************************************************************
          IREC     DS    0CL18         1-18       Offer record
          ISEM     DS    CL3           1- 3       Semester
          ICID     DS    0CL5          4- 8       Course ID
          IDEPT    DS    CL2           4- 5       Department
                   DS    CL3           6- 8       Course number
          ISECT    DS    CL1           9- 9       Section number
          ITID     DS    CL3          10-12       Teacher ID
          IROOM    DS    CL4          13-16       Room number
          IOCRLF   DS    CL2          17-18       PC/370 only - CR/LF
          *****************************************************************
          *        Output (line) definition                               *
          *****************************************************************
          OREC     DS    0CL42         1-42
                   DC    CL10' '       1-10
          ODEPT    DS    CL2          11-12       Department
                   DC    CL7' '       13-19
          O#DEPT   DS    CL4          20-23       Number of sections
                   DC    CL17' '      24-40
```

```
OCRLF    DS    CL2          41-42         PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
*          * --+--1--+--2--+--3--+--4
*            COURSE OFFERINGS              PageBZZ9
*              Semester XXX
*          *            Dept     Sections
*           ---    ----
*            XX        BZZ9
*            XX        BZZ9
*           ---    ----
*            Total     BZZ9
*
FORMFEED DS    0CL42                     PC/370 only
*        DC    X'0C'                     EBCDIC formfeed
*        DC    CL39' '
         DC    40C'_'                    For testing...
         DC    X'0D25'                   EBCDIC CR/LF
HD1      DS    0CL42
         DC    CL36'               COURSE OFFERINGS                    Page'
HDPGS    DC    CL4'BZZ9'
         DC    XL2'0D25'
HD2      DS    0CL42
         DC    CL21'                  Semester '
HDSEM    DS    CL3
         DC    CL16' '
         DC    XL2'0D25'
HD3      DS    0CL42
         DC    CL40' '
         DC    XL2'0D25'
HD4      DS    0CL42
         DC    CL40'          Dept   Sections           '
         DC    XL2'0D25'
HD5      DS    0CL42
         DC    CL40'        ---  ----
         DC    XL2'0D25'
         END   BEGIN
```

10.6. Ejercicios

1. 1. Verdadero o falso.

   a) En el procesamiento de romper el control, el campo de romper el control (s) se inicializan en el SETUP F T antes de la imprimación leer.

   b) Cuando se rompe el control de múltiples niveles se utilizan, la ruptura de control de los campos se F T evaluados menores antes de los principales.

   c) Cuando se rompe el control de múltiples niveles se utilizan, una importante ruptura implica un menor de edad F T romper.

   d) d. En lugar de los contadores de incremento en todos los niveles dentro del proceso, los totales de los principales F T se puede "rodar" en los totales de menor importancia.

   e) Cuando un periodo de control se detecta, contadores o los totales se deshacen antes de que se F T cero.

   f) Múltiples control de procesamiento a nivel de ruptura requiere dos campos implican para cada nivel. F T

   g) Una técnica simple para forzar un salto de página en el procesamiento de control de descanso es F T establecer la página contra el máximo permitido.

   h) Adición de romper la lógica de control para un programa requerirá un cambio en la F T la línea principal.

   i) Adición de romper la lógica de control para un programa requerirá un cambio en la F T Rutina Wrapup.

   j) Adición de romper la lógica de control para un programa requerirá un cambio en el informe de F T DCB.

   k) Los cambios en los campos de la ruptura de control son revisados antes de formato del informe F T línea de detalle para el registro actual.

   l) Romper la lógica de control puede ser utilizado en los informes resumidos, así como en los informes de los detalles. F T

   m) Un contador de línea separada es necesaria para cada nivel de procesamiento de control de descanso. F T

2. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Elaborar una lista de las herramientas encontradas en cada kit. Permita que 10 líneas por página. Su resultado debería aparecer como siguiente:

```
.         1         2         3         4
12345678901234567890123456789012345678901
```

```
SMALL TOWN HARDWARE       Page BZZ9
 Kit ID    Tool ID
 ------    -------
  XXX        XXX
             XXX
  XXX        XXX
             XXX
             XXX
```

El kit de identificación deben imprimirse en la primera línea para que el kit, y en la primera línea de una página si esa lista debe kit de un periodo de duración de página. Doble espacio entre los equipos.

3. (Consulte la base de datos Small Town Banco de Sangre en más conjuntos de datos.) Uso de la donación de archivo presentar un resumen de la historia del grupo de la siguiente manera:

```
.         1         2         3
12345678901234567890123456789
    SMALL TOWN BLOOD BANK
               Number of
    Group ID   Donations
    --------   ---------
      XXX        BZZ9
      XXX        BZZ9
      XXX        BZZ9
    --------   ---------
     Total       BZZ9
```

Con el fin de elaborar este informe, el archivo tendrá que ser ordenados por ID de grupo. SORT DOS ' comando es crudo, pero será suficiente. Escriba lo siguiente en el símbolo de DOS:

```
sort /+4 < donation.dat > donation.srt
```

Esto creará un nuevo archivo, DONATION.SRT, que se ordena a partir de la cuarta de columna de cada registro. Especifique DONATION.SRT como la DDNAME de la DCB de entrada.

4. (Refer to the Small Town Blood Bank database in More Datasets.) Produce a donor history summary as follows:

```
.        1         2         3         4         5
123456789012345678901234567890123456789012345678901
              SMALL TOWN BLOOD BANK       Page BZZ9
                  First      Last     Number of
     Donor ID   Donation   Donation   Donations
     --------   --------   --------   ---------
```

```
  XXX       mm/dd/yy    mm/dd/yy       BZZ9
  XXX       mm/dd/yy    mm/dd/yy       BZZ9
  XXX       mm/dd/yy    mm/dd/yy       BZZ9
--------    --------    --------    ---------
 Total                                 BZZ9
```

In order to produce this report, the file will need to be sorted by donor ID. DOS' SORT command is crude but will suffice. Type the following at the DOS prompt:

```
sort /+1 < donation.dat > donation.srt
```

This will create a new file, DONATION.SRT , which is sorted beginning with the first column of each record. Specify DONATION.SRT as the DDNAME for the input DCB .

(Hint: reformat the dates from mmddyy to yymmdd . The minimum date for a particular donor will be the first donation date, and the maximum date will be the last donation date.)

5. (Refer to the Small Town Payroll database in More Datasets.) Use the HISTORY file to produce a payroll register for all pay periods. The report should appear as follows, with one pay period per page :

```
.        1         2         3         4         5
12345678901234567890123456789012345678901234567890
            SMALL TOWN PAYROLL
     Payroll Register for PPED mm/dd/yy
     Employee    Hours         Gross
     --------   -------    -----------
       XXX      BZZ9.99    BZZ,ZZ9.99
       XXX      BZZ9.99    BZZ,ZZ9.99
       XXX      BZZ9.99    BZZ,ZZ9.99
     --------   -------    -----------
      TOTAL     BZZ9.99    BZZ,ZZ9.99
There were BZZ9 checks printed for PPED mm/dd/yy.
```

Capítulo 11

Más de una entrada de archivo

Objetivos Al término de este capítulo usted será capaz de: Ordenar datos

- archivos según sea necesario para procesar utilizando lógica de concordancia y
- Diseño y programación de un programa de uso de la lógica concordancia con proceso de dos archivos de entrada de forma simultánea.

Introducción Todos los programas que hemos visto hasta este momento han tenido sólo un archivo de entrada y una archivo de salida. Los dos capítulos siguientes se verá en la lógica necesaria para procesar dos archivos de entrada. Cuándo procesamiento de dos archivos de entrada, por lo general se puede pensar en uno de los archivos como un archivo maestro y el otro como un archivo de transacciones. La naturaleza de las operaciones puede o no puede requerir algunos cambios en el registro correspondiente en el archivo maestro. En este capítulo vamos a proceso en dos archivos, pero no vamos a realizar cambios en el archivo maestro: simplemente elaborar un informe haciendo uso de los datos contenidos en el ambos archivos. En el próximo capítulo vamos a aplicar los cambios en el archivo maestro como resultado de la transacciones procesadas.

* *

El informe que vamos a producir en este capítulo es una lista de tareas de enseñanza para el otoño de 1992 semestre. El informe aparecerá como sigue:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
            F92 Teaching Assignments        PageBZZ9
TID         Name          Course   Section   Room
---  ---------------    ------   -------   ----
```

```
XXX  XXXXXXXXXXXXXXX    XXXXX       X      XXXX
                         XXXXX       X      XXXX
XXX  XXXXXXXXXXXXXXX    XXXXX       X      XXXX
                         XXXXX       X      XXXX
```

Observe que aparecerá una lista de ID y el nombre del profesor para cada profesor de enseñanza de un curso en el otoño de Semestre de 1992, junto con el ID del curso, número de sección y número de habitación para cada curso enseñado. El profesor ID, ID del curso, el número de sección y número de habitación vendrá de la oferta del archivo, pero el nombre del maestro se encuentra en el archivo sólo MAESTRO! Podemos pensar en el archivo como MAESTRO el maestro y el archivo de oferta como de las transacciones.

Lo que necesitamos es de alguna manera de encontrar el nombre de los maestros basados en el maestro de identificación. Por supuesto, este es un tarea trivial si se está trabajando con un sistema de base de datos relacional, pero no lo son. Debemos código esta lógica concordancia nosotros mismos. Demostrar que la lógica es el propósito de este capítulo. Nota: no introducir nuevas instrucciones de ensamblador en este capítulo.

11.1. Clasificación de los archivos de entrada

Hemos dicho que el profesor ID, ID del curso, número de sección y número de habitación vendrá desde el archivo de oferta, y el nombre del maestro provienen del archivo MAESTRO. Para imprimir la nombre de los maestros basados en el maestro de identificación, que se coinciden con los dos archivos basado en el maestro de identificación. Sort El profesor de identificación se conoce como el campo clave. Con el fin de hacerlo, ambos archivos deben estar en maestro de De secuencia de identificación, es decir, deben ser ordenados. Tipo de comando de DOS "es cruda, pero será suficiente. (En el mundo mainframe que usaría un mucho más sofisticado paquete de clase, como como Syncsort.) Para ordenar estos archivos, que escriba el siguiente en el símbolo de DOS:

```
sort /+10 < offer.dat > offer.srt
sort /+1 < teacher.dat > teacher.srt
```

Estas instrucciones ordenar los archivos. DAT, Informe.txt la creación de los archivos. SRT.

Podemos utilizar DOS ' Comando TYPE para ver estos archivos. Para ejemplo:

```
A:\MIN>type offer.srt A:\MIN>type teacher.srt
                      218HINCKLEY, G.B. MBA N5509
W92MA1071218A2                          574SMITH, J. F92AC1011218B1
                      626YOUNG, B. F92BU1011218B1                                MBA Y5664
                      732BENSON, E.T. W92PE1511574GYM                               PHD N5156
                      854KIMBALL, S.W. PHD Y5594
```

```
F92PE1511574GYM
W93PE1511574GYM
F92MA1011626A2
F92MA1012626A2
W93MA1011626A2
W93MA1071626A3
W92EG1021732A1
F92EG1011732A1
F92EG1012732A1
W93EG1021854A1
```

Estos archivos. SRT será incluido en el parámetro de la DCB DDNAME en el programa:

```
TEACHERS DCB LRECL=29,RECFM=F,MACRF=G,EODAD=ATENDTCH,
             DDNAME='TEACHER.SRT'
OFFER    DCB LRECL=18,RECFM=F,MACRF=G,EODAD=ATENDOFF,
             DDNAME='OFFER.SRT'
```

11.2. La estructura de la línea principal

La estructura de la línea principal del programa es realmente no muy diferente de lo que hemos visto antes: seguimos proceso de registros hasta que o bien el archivo está en EOF. Este tipo de cheque requieren de dos de fin de archivo de interruptores:

```
EOFTEACH DC CL1'N'
EOFOFFER DC CL1'N'
```

Tenga en cuenta que es posible (y de hecho probable) que ambos archivos no llegará EOF, al mismo tiempo. (Esta condición se maneja en el Wrapup de rutina.)

Wrapup El código de la línea principal se repite aquí. No es en particular, "limpio" el código en comparación con el diagrama de flujo, pero esto es porque no BAL FIN tienen una estructura o SI

260 ⟨*search..., main loop SETUP,EOFTEACH,EOFOFFER to EOJ,PROCESS,WRAPUP* 260⟩≡ (266)

```
          BAL    R10,SETUP
MAIN      EQU    *
          CLI EOFTEACH,C'Y'
          BE  EOJ
          CLI EOFOFFER,C'Y'
          BE  EOJ
          BAL    R10,PROCESS
          B      MAIN
EOJ       EQU    *
          BAL    R10,WRAPUP
```

11.3. La rutina de instalación

Dentro de la rutina de instalación, abrimos todos los archivos y de leer el primer registro de cada fichero de entrada:

```
SETUP EQU  *
      ST   R10,SVSETUP
      OI   TEACHERS+10,X'08'
      OI   OFFER+10,X'08'
      OI   REPORT+10,X'08'
      OPEN TEACHERS
      OPEN OFFER
      OPEN REPORT
      BAL  R10,READTCH
      BAL  R10,READOFF
      L    R10,SVSETUP
      BR   R10
```

11.4. Las rutinas de LEA

En todos los programas que hemos visto hasta ahora, hemos ha tenido una rutina de lectura, un extremo de archivo MAESTRO interruptor, una entrada DCB, etc En este programa, Nosotros, por supuesto, tienen dos de cada uno. El las rutinas de lectura se muestran a continuación. Ellos son GET muy similar con la excepción de que desde MAESTRO ACTA estamos elaborando una lista de clase para el semestre de F92 sólo, vamos a saltar todos los registros OFERTA no desde el semestre:

```
READTCH  EQU *
         ST  R10,SVREADT
         GET TEACHERS,TREC
         B   READTX
ATENDTCH EQU *
         MVI EOFTEACH,C'Y'
READTX   EQU *
         L   R10,SVREADT
         BR  R10
READOFF  EQU *
         ST  R10,SVREADO
READOFF2 EQU *
         GET OFFER,OREC
         CLC OSEM,=CL3'F92'
         BNE READOFF2
         B   READOX
ATENDOFF EQU *
         MVI EOFOFFER,C'Y'
```

```
READOX   EQU *
         L   R10,SVREADO
         BR  R10
```

11.5. El proceso de rutina PROCESO

ID maestro de los maestros En el momento en que realizamos el archivo superior Rutina de proceso, tenemos un solo Maestro ID de archivo OFRECEMOS? registro de cada archivo. A continuación, comparar el ID maestro de maestro de los maestros de identificación los dos archivos. Hay tres archivos menos de Maestro ID del archivo OFRECEMOS? condiciones posibles como resultado OFERTA BAJA de la presente comparar:

- la ID en el archivo MAESTRO
- MAESTRO es mayor que la ID en MATCH BAJA el archivo OFERTA, la ID en el archivo MAESTRO es menor que la ID en el Archivo de oferta, o la ID en el archivo MAESTRO
- es igual a la ID en el Archivo OFERTA. VOLVER

La primera condición podría indicar que tenemos un Registro de oferta con un profesor de identificación que no se encuentra en el archivo MAESTRO. Evidentemente, esto podría indicar un error condición.

La segunda condición que podría indicar que tenemos un TTiD Registro maestro con un profesor de identificación que no se encuentra en el archivo OFERTA. Esto indicaría que este profesor particular no es la enseñanza de los cursos durante el semestre F92. (Simplemente, se saltará estos R10 BAL,

La tercera condición que indique que tenemos un Registro de oferta con un profesor de identificación que se encuentra en el archivo. Esto indica que este profesor y curso deben ser listados en el informe.

Cada una de estas condiciones serán tratados por separado en un de rutina.

El "Maestro de archivo bajo" condición indica que tenemos un TEACHLOW maestro que no es la enseñanza de las clases este semestre. No el procesamiento es necesario; acaba de obtener el registro de profesor siguiente.

```
TEACHLOW EQU *
ST  R10,SVTCHLOW
BAL R10,READTCH
L   R10,SVTCHLOW
BR  R10
```

La "Oferta de archivo bajo" condición indica que tenemos un curso de Mover Maestro ID de la oferta no asignado a un maestro en el archivo MAESTRO. Vamos a seguir adelante de archivo con ID de y mostrar el curso de todos modos, y mostrar el nombre del profesor como "Desconocido".

```
OFFERLOW EQU *
         ST  R10,SVOFFLOW
         MVC HOLDTID,OTID
         MVC HOLDNAME,=CL15'Unknown'
         BAL R10,DOTHIS1
         L   R10,SVOFFLOW
         BR  R10
```

El "Match" condición indica que tenemos un curso de Profesor de archivo de asignado a un profesor que está en el archivo MAESTRO. Hold proceso de identificación todos los registros de curso de este maestro.

```
MATCH EQU *
      ST  R10,SVMATCH     Teacher file to Hold
      MVC HOLDTID,TTID          Name
      MVC HOLDNAME,TTNAME
      BAL R10,DOTHIS1
      BAL R10,READTCH
      L   R10,SVMATCH
      BR  R10
```

Ahora todo el proceso la oferta de cursos de un maestro único de identificación. En el OFFERLOW y MATCH rutinas pasamos el ID maestro y el nombre del maestro (o "Desconocido") a una zona de bodegas. Estos campos zona de la bodega servir a dos propósitos. En primer lugar, la bodega de ID se utiliza para comprobar si hay cambios en el maestro de identificación. En segundo lugar, estos son los campos que se trasladó a la identificación y el nombre del maestro en el informe. Vamos a mostrar la ID maestro y el nombre en la primera línea para este único maestro, a menos que la lista de cursos para este profesor debería ocurrir con "span" un salto de página, en cuyo caso la ID y el nombre del profesor será mostrado en de la primera línea de la página siguiente también.

```
DOTHIS1 EQU *
        ST  R10,SVTHIS1
        MVC RREC,BLANKS
        MVC RTID,HOLDTID
        MVC RTNAME,HOLDNAME
DOTHIS2 EQU *
        BAL R10,CHKLNS
        CP  LNS,=P'0'
        BNE DOTHIS3
        MVC RTID,HOLDTID
        MVC RTNAME,HOLDNAME
        DOTHIS3 EQU *
        MVC RCID,OCID
        MVC RSECT,OSECT
        MVC RROOM,OROOM
```

```
        BAL R10,WRITE
        BAL R10,READOFF
        CLI EOFOFFER,C'Y'
        BE  DOTHIS4
        CLC HOLDTID,OTID
        BE  DOTHIS2
        MVC RREC,BLANKS
        BAL R10,WRITE
        L   R10,SVTHIS1
        BR  R10
```

Se realiza una comprobación de una página completa antes de cada registro que se está impreso (ver CHKLNS). En el proceso de el formato de una línea informe, se compruebe si el recuento de línea (LNS) es cero. Si es así, entonces esta es la primera línea en una página: mostrar el ID maestro y nombre de nuevo. Después de cada línea tiene el formato y el impreso, el siguiente oferta de cursos registro que se lee. El proceso se repite hasta que la oferta de cursos de archivos es un EOF o el maestro ID ha cambiado (en comparación con la bodega de identificación).

11.5.1. El CHKLNS, HDGS, y rutinas de escritura

El CHKLNS, HDGS, y las rutinas de escribir son las mismas que hemos visto antes:

```
CHKLNS  EQU *
        ST  R10,SVCHKLNS
        CP  LNS,MAXLNS
        BL  CHKLNSX
        BAL R10,HDGS
CHKLNSX EQU *
        L   R10,SVCHKLNS
        BR  R10
```

.

```
HDGS EQU *
     ST  R10,SVHDGS
     AP  PGS,=P'1'
     MVC HDPGS,=X'40202120'
     ED  HDPGS,PGS
     PUT REPORT,FORMFEED
     PUT REPORT,HD1
     PUT REPORT,HD2
     PUT REPORT,HD3
     PUT REPORT,HD4
     ZAP LNS,=P'0'
```

```
                    L   R10,SVHDGS
                    BR  R10
```

.

```
WRITE EQU *
      ST  R10,SVWRITE
      PUT REPORT,RREC
      AP  LNS,=P'1'
      L   R10,SVWRITE
      BR  R10
```

11.6. La rutina Wrapup

Recuerde que en nuestro análisis de la lógica de la línea principal que seguirá el proceso hasta que los registros o bien el archivo está en EOF, y que es probable que ambos archivos no llegará EOF, al mismo tiempo. Nosotros debe, por tanto, tener en cuenta que cuando llegamos a Wrapup, es probable que uno de los archivos no se encuentra en final del archivo. Vamos a ejecutar uno de los siguientes ciclos: Si el archivo de oferta no es en EOF, el proceso de todos los registros restantes del curso ofrendas como inigualable;

- es decir, OFFERLOW. De lo contrario, si el archivo maestro no está a EOF, el proceso de todos los registros restantes como maestro
- sin precedentes, es decir, TEACHLOW.

```
WRAPUP  EQU   *
        ST    R10,SVWRAP
WRAPUP2 EQU   *
        CLI   EOFOFFER,C'Y'
        BE    WRAPUP3
        BAL   R10,OFFERLOW
        B     WRAPUP2
WRAPUP3 EQU   *
        CLI   EOFTEACH,C'Y'
        BE    WRAPUP4
        BAL   R10,TEACHLOW
        B     WRAPUP3
WRAPUP4 EQU   *
        CLOSE TEACHERS
        CLOSE OFFER
        CLOSE REPORT
        WTO   'TEACH11A ... Teacher list on REPORT.TXT'
        L     R10,SVWRAP
        BR    R10
```

11.6.1. La salida del programa

```
          F92 Teaching Assignments     Page 1
TID       Name          Course  Section Room
--- ---------------   ------  ------- ----
218 HINCKLEY, G.B.    AC101       1   B1
                      BU101       1   B1
574 SMITH, J.         PE151       1   GYM
626 YOUNG, B.         MA101       1   A2
          F92 Teaching Assignments     Page 2
TID       Name          Course  Section Room
--- ---------------   ------  ------- ----
626 YOUNG, B.         MA101       2   A2
732 BENSON, E.T.      EG101       1   A1
                      EG101       2   A1
```

11.7. Programa de Solución

El programa completa y comentada, TEACH11A.MLC, sigue. En este programa, MAXLNS se estableció en 6, por lo de que al menos un maestro (joven) que "span" un salto de página. Esto se hizo con el fin de demostrar que el nombre de un profesor se muestra en la parte superior de la página si los cursos que un maestro abarcó un salto de página.

266 ⟨*TEACH11A.MLC* 266⟩≡

```
                PRINT NOGEN
*****************************************************************
*        FILENAME: TEACH11A.MLC                                 *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce list of F92 teaching assignments.    *
*                     This program illustrates matching logic.  *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'TEACH11A ... Begin execution'
⟨search..., main loop SETUP,EOFTEACH,EOFOFFER to EOJ,PROCESS,WRAPUP 260⟩
         WTO    'TEACH11A ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                  before any records are processed.            *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
```

```
                OI      TEACHERS+10,X'08' PC/370 ONLY - Convert all
*                                                 input from ASCII to EBCDIC
                OI      OFFER+10,X'08'            PC/370 ONLY - Convert all
*                                                 input from ASCII to EBCDIC
                OI      REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                                 output from EBCDIC to ASCII
                OPEN TEACHERS
                OPEN OFFER
                OPEN REPORT
                BAL     R10,READTCH
                BAL     R10,READOFF
                L       R10,SVSETUP
                BR      R10
***************************************************************
*               HDGS - Print headings.                        *
***************************************************************
HDGS            EQU   *
                ST    R10,SVHDGS
                AP    PGS,=P'1'                 Add 1 to page count
                MVC   HDPGS,=X'40202120' Edit pattern for page count
                ED    HDPGS,PGS                 Move page count to heading
                PUT   REPORT,FORMFEED           PC/370 ONLY
                PUT   REPORT,HD1
                PUT   REPORT,HD2
                PUT   REPORT,HD3
                PUT   REPORT,HD4
                ZAP   LNS,=P'0'                 Reset line count to zero
                L     R10,SVHDGS
                BR    R10
***************************************************************
*               PROCESS - Those things which happen once per record.  *
***************************************************************
PROCESS EQU     *
                ST    R10,SVPROC
                CLC   TTID,OTID                 Attempt match on teacher ID
                BH    PROC2                     Offerings low
                BL    PROC3                     Teacher low
                BAL   R10,MATCH                 Otherwise a match was found
                B     PROCESSX
PROC2           EQU   *                         No teacher for this offering
                BAL   R10,OFFERLOW
                B     PROCESSX
PROC3           EQU   *                         No offerings for this teacher
                BAL   R10,TEACHLOW
PROCESSX EQU    *
                L     R10,SVPROC
```

```
         BR    R10
*****************************************************************
*        TEACHLOW - No courses for this teacher.                *
*                   This is NOT an error condition.             *
*                   Just skip this teacher and go to next.      *
*****************************************************************
TEACHLOW EQU   *
         ST    R10,SVTCHLOW
         BAL   R10,READTCH              Read next teacher record
         L     R10,SVTCHLOW
         BR    R10
*****************************************************************
*        MATCH - Course offering with teacher.                  *
*****************************************************************
MATCH    EQU   *
         ST    R10,SVMATCH
         MVC   HOLDTID,TTID             Current teacher
         MVC   HOLDNAME,TTNAME
         BAL   R10,DOTHIS1              Do this one (teacher)
         BAL   R10,READTCH              Read next teacher record
         L     R10,SVMATCH
         BR    R10
*****************************************************************
*        OFFERLOW - Course offering without matching teacher    *
*****************************************************************
OFFERLOW EQU   *
         ST    R10,SVOFFLOW
         MVC   HOLDTID,OTID             Current teacher
         MVC   HOLDNAME,=CL15'Unknown'
         BAL   R10,DOTHIS1              Do this one (teacher)
         L     R10,SVOFFLOW
         BR    R10
*****************************************************************
*        DOTHIS1 - Do this one (teacher)                        *
*****************************************************************
DOTHIS1 EQU    *
         ST    R10,SVTHIS1
         MVC   RREC,BLANKS
         MVC   RTID,HOLDTID             First line for each teacher
         MVC   RTNAME,HOLDNAME            will show ID and name
DOTHIS2 EQU    *                       This loop for each offering
         BAL   R10,CHKLNS                 for this teacher...
         CP    LNS,=P'0'
         BNE   DOTHIS3
         MVC   RTID,HOLDTID             Also shown on first line
         MVC   RTNAME,HOLDNAME            of each page
```

```
          DOTHIS3 EQU    *
                  MVC    RCID,OCID
                  MVC    RSECT,OSECT
                  MVC    RROOM,OROOM
                  BAL    R10,WRITE
                  MVC    RREC,BLANKS
                  BAL    R10,READOFF              Read next offerings record
                  CLI    EOFOFFER,C'Y'            If EOF then done
                  BE     DOTHIS4
                  CLC    HOLDTID,OTID             Still the same teacher?
                  BE     DOTHIS2                    Yes, continue this one
          DOTHIS4 EQU    *
                  MVC    RREC,BLANKS              Blank line between teachers
                  BAL    R10,WRITE
                  L      R10,SVTHIS1
                  BR     R10
          ***************************************************************
          *        READTCH - Read a teacher record.                     *
          ***************************************************************
          READTCH EQU    *
                  ST     R10,SVREADT
                  GET    TEACHERS,TREC            Read a single teacher record
                  B      READTX
          ATENDTCH EQU   *
                  MVI    EOFTEACH,C'Y'
          READTX  EQU    *
                  L      R10,SVREADT
                  BR     R10
          ***************************************************************
          *        READOFF - Read a course offerings record.            *
          ***************************************************************
          READOFF EQU    *
                  ST     R10,SVREADO
 READOFF2 EQU   *
                  GET    OFFER,OREC                Read a single offerings record
                  CLC    OSEM,=CL3'F92'            Process F92 records only
                  BNE    READOFF2
                  B      READOX
          ATENDOFF EQU   *
                  MVI    EOFOFFER,C'Y'
          READOX  EQU    *
                  L      R10,SVREADO
                  BR     R10
 ***************************************************************
 *        CHKLNS - Check lines printed. Full page?             *
 ***************************************************************
```

⟨ *search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS* 184 ⟩

```
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,RREC               Write report line
         AP    LNS,=P'1'
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
*                                        At this point we know that
*                                        at least one of the input
*                                        files is at EOF. Process
*                                        other file as "unmatched"
*                                        until at EOF also.
WRAPUP2 EQU    *
         CLI   EOFOFFER,C'Y'
         BE    WRAPUP3
         BAL   R10,OFFERLOW
         B     WRAPUP2
WRAPUP3 EQU    *
         CLI   EOFTEACH,C'Y'
         BE    WRAPUP4
         BAL   R10,TEACHLOW
         B     WRAPUP3
WRAPUP4 EQU    *
         CLOSE TEACHERS
         CLOSE OFFER
         CLOSE REPORT
         WTO   'TEACH11A ... Teacher list on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*************************************************************
*        Literals, if any, will go here                     *
*************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATENDTCH,
```

```
                DDNAME='TEACHER.SRT'
OFFER    DCB    LRECL=18,RECFM=F,MACRF=G,EODAD=ATENDOFF,
                DDNAME='OFFER.SRT'
REPORT   DCB    LRECL=62,RECFM=F,MACRF=P,
                DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP DC      F'0'                    SETUP
SVHDGS   DC     F'0'                    HDGS
SVPROC   DC     F'0'                    PROCESS
SVREADT DC      F'0'                    READTCH
SVREADO DC      F'0'                    READOFF
SVWRITE DC      F'0'                    WRITE
SVWRAP   DC     F'0'                    WRAPUP
SVCHKLNS DC     F'0'                    CHKLNS
SVMATCH DC      F'0'                    MATCH
SVOFFLOW DC     F'0'                    OFFERLOW
SVTCHLOW DC     F'0'                    TEACHLOW
SVTHIS1 DC      F'0'                    DOTHIS1
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC     X'0D25'                 PC/370 ONLY - EBCDIC CR/LF
EOFTEACH DC     CL1'N'                  End of teacher file? (Y/N)
EOFOFFER DC     CL1'N'                  End of offerings file? (Y/N)
PGS      DC     PL2'0'                  Nbr of pages printed.
LNS      DC     PL2'6'                  Lines printed on this page.
MAXLNS   DC     PL2'6'                  Max nbr lines per page.
*                                       My line counts exclude hdgs.
BLANKS   DS     0CL62
         DC     CL60' ',XL2'0D25'
HOLDTID DC      CL3' '                  Hold (current) teacher ID
HOLDNAME DC     CL15' '                 Hold (current) teacher name
*****************************************************************
*        Input record definition - Teacher                      *
*****************************************************************
TREC     DS     0CL29          1-29     Teacher record
TTID     DS     CL3            1- 3     Teacher ID nbr
TTNAME   DS     CL15           4-18     Teacher name
TTDEG    DS     CL4           19-22     Highest degree
TTTEN    DS     CL1           23-23     Tenured?
TTPHONE DS      CL4           24-27     Phone nbr
TTCRLF   DS     CL2           28-29     PC/370 only - CR/LF
*************************************************************
*        Input record definition - Offerings                *
```

```
*****************************************************************
OREC     DS    0CL18       1-18      Offerings record
OSEM     DS    CL3         1- 3      Semester
OCID     DS    CL5         4- 8      Course ID
OSECT    DS    CL1         9- 9      Section number
OTID     DS    CL3        10-12      Teacher ID
OROOM    DS    CL4        13-16      Room number
OOCRLF   DS    CL2        17-18      PC/370 only - CR/LF
*****************************************************************
*        Report (line) definition                               *
*****************************************************************
RREC     DS    0CL62       1-62      Report record
RTID     DS    CL3         1- 3      Teacher ID nbr
         DC    CL2' '      4- 5
RTNAME   DS    CL15        6-20      Teacher name
         DC    CL3' '     21-23
RCID     DS    CL5        24-28      Course ID
         DC    CL6' '     29-34
RSECT    DS    CL1        35-35      Section number
         DC    CL5' '     36-40
RROOM    DS    CL4        41-44      Room number
         DC    CL16' '    45-60
RCRLF    DS    CL2        61-62      PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
FORMFEED DS    0CL62                 PC/370 only
*        DC    X'0C'                 EBCDIC formfeed
*        DC    CL59' '
         DC    60C'_'                For testing...
         DC    X'0D25'               EBCDIC CR/LF
HD1      DS    0CL62
         DC    CL40'            F92 Teaching Assignments      '
         DC    CL4'Page'       HDPGS    DC    CL4'BZZ9'
         DC    CL12' '                DC    XL2'0D25'
HD2      DS    0CL62                DC    CL60' '
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL40'TID         Name           Course Section '
         DC    CL20'Room'
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'-- --------        --- ---- '
         DC    CL20'--'
         DC    XL2'0D25'
         END   BEGIN
```

11.8. Ejercicios

1. 1. Verdadero o falso. Cuando dos archivos se procesan utilizando la lógica de juego ...

   a) un archivo general, se puede considerar como el archivo maestro y el otro como el F T archivo de transacciones.

   b) ambos archivos deben ser secuenciados en el campo clave. F T

   c) Tipo de comando de DOS "crea un nuevo archivo que contiene los registros ordenados. F T

   d) el parámetro de la

   ```
   DCB DDNAME
   ```

   debe referirse al archivo ordenado. F T

   e) de la lógica de largo recorrido continuará hasta que ambos archivos se encuentran en EOF. F T

   f) la rutina de instalación contendrá un oscurecimiento de lectura para ambos archivos. F T

   g) LEA será la rutina de leer un registro de cada archivo. F T

   h) PROCESO compara la rutina de un registro de cada archivo basado en el primer campo de F T en el registro.

   i) hay dos condiciones posibles como resultado de la comparación en el PROCESO F T de rutina.

   j) registros no coincidentes siempre indica un error fatal. F T

   k) habrá un CHKLNS separado y HDGS rutinas para cada archivo de entrada. F T

   l) se lee el siguiente registro de cada archivo al final de la rutina PROCESO. F T

   m) cuando llegamos a Wrapup, sabemos que ambos archivos se encuentran en EOF. F T

2. ¿Qué cambios haría usted a TEACH11A.MLC para mover el cheque por F92 semestre los registros de la READOFF de rutina a la rutina PROCESO? (Mostrar las dos rutinas.)

3. (Consulte el auto de pueblo pequeño de almacenamiento en la base de datos más conjuntos de datos.) Elaborar una lista de inquilinos casillero de la siguiente manera:

```
.        1         2         3         4         5
12345678901234567890123456789012345678901234567890
SMALL TOWN self-storage Page BZZ9
Inquilinos actual
```

```
Atención al cliente Locker
------------------------- --------
Nombre ID # No Tipo pagados a través de
----- ------------------- - ---- ---------
XXXXX XXXXXXXXXXXXXXXXXXX XX X mm / dd / aa
XXXXX XXXXXXXXXXXXXXXXXXX XX X mm / dd / aa
XXXXX XXXXXXXXXXXXXXXXXXX XX X mm / dd / aa
```

Usted necesitará el CLIENTE y los archivos de taquilla. Con el fin de elaborar este informe, ambos archivos deben ser en la secuencia de identificación de cliente: escriba lo siguiente en el símbolo de DOS:

```
Ordenar / 1 <cust.dat> cust.srt
Ordenar / 4 <locker.dat> locker.srt
```

Especifique CUST.SRT y LOCKER.SRT como la DDNAME para la DCB de entrada.

4. (Consulte el auto de pueblo pequeño de almacenamiento en la base de datos más conjuntos de datos.) Producir un armario informe de síntesis de la disponibilidad de la siguiente manera:

```
.        1         2         3         4
12345678901234567890123456789012345678901
                        SMALL TOWN self-storage
                     Locker Disponibilidad Resumen
------ ---------- ------ ---------
X XXXXXXXXX BZZ9.99 BZZ9
X XXXXXXXXX BZZ9.99 BZZ9
X XXXXXXXXX BZZ9.99 BZZ9
```

Usted necesitará la taquilla y archivos con una velocidad. Con el fin de elaborar este informe, ambos archivos deben ser en la secuencia de tipo Locker: escriba lo siguiente en el símbolo de DOS:

```
Ordenar / 3 <locker.dat> locker.srt
Ordenar / 1 <rate.dat> rate.srt
```

Especifique LOCKER.SRT y RATE.SRT como la DDNAME para la DCB de entrada.

Nota: Este es un resumen , el informe de uso de la lógica de control de descanso. Usted no está lista cada casillero disponible, sólo el número de de taquillas disponibles de cada tipo. Un armario se considera disponible si el ID de cliente en el LOCKER el archivo está en blanco.

5. (Consulte la base de datos Small Town Banco de Sangre en más conjuntos de datos.) Producir un donante de resumen de la historia de la siguiente manera:

```
.        1         2         3         4         5
12345678901234567890123456789012345678901234567890
SMALL TOWN BANCO DE SANGRE Page BZZ9
Número de Apellidos de
Donante ID Nombre del Donante Donación Donación Donaciones
-------- ------------ -------- -------- ---------
XXX X XXXXXXXXXX mm / dd / mm / aa / dd / aa BZZ9
XXX X XXXXXXXXXX mm / dd / mm / aa / dd / aa BZZ9
XXX X XXXXXXXXXX mm / dd / mm / aa / dd / aa BZZ9
-------- ------------ -------- -------- ---------
                                        BZZ9 total
```

Nota: Este programa es similar al ejercicio 4 del capítulo 10, salvo que el nombre de los donantes ha ha añadido al informe. Vea las instrucciones adicionales en ese capítulo. LA DONACIÓN archivo deben ser ordenados como se muestra allí. Este programa también leer el archivo de donantes, pero ese archivo no deben ser ordenados como ya se hace en la secuencia de identificación de donantes. Mostrar la primera inicial y el último nombre de los donantes.

6. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Utilizar las ventas de archivo Para actualizar la cantidad en el campo contrario en el archivo de TOOL. La cantidad por la mano (TQOH) se reducido por la cantidad vendida (TSOLD). No actualizar la versión actualizada de toda otra fields.Write TOOL archivo NEWTOOL.DAT. (Usted no se ha elaborado un informe.)

   Si una herramienta de identificación se encuentra en las ventas archivo sin un instrumento de correspondencia ID en el archivo de herramienta, luego de la WTO un mensaje apropiado y el registro. (Usted tendrá que hacer otra versión de SALES.DAT para poner a prueba esta parte de su lógica.)

   No hay necesidad de ordenar los datos, ya que ambos expedientes ya se han secuenciado por la herramienta de identificación.

Capítulo 12

cuencial del archivo de actualización

Objetivos Al término de este capítulo usted será capaz de:

- Lista de los tres tipos de transacción utilizados en un programa secuencial de archivo de actualización,
- Dado MASTER simulada y archivos de transacciones, determinar el contenido de la nueva MASTER archivo,
- Dado MASTER simulados y archivos de transacciones, determinar el contenido del informe de auditoría y de conciliar los cargos que figuran en él,
- Describir la transformación necesaria para MASTER LOW, transacción, y acordes con las condiciones en un secuencial de programa de actualización de archivos, y
- Diseño y programación de un programa de actualización que utiliza la lógica para aplicar las transacciones a un archivo maestro.

Introducción BQSFUMST.DAT BQSFUTRN.DAT (Ya en el cliente (Ya en el cliente En el capítulo anterior vimos el Número de secuencia.) Número de secuencia.) lógica necesaria para procesar dos archivos de entrada. Los programas que se crean los informes que contienen TRANS-VIEJO los datos de ambos archivos. La lógica de juego ACCIONES MASTER discuten allí es común a muchos aplicaciones de negocio. En este capítulo dar ese paso lógico uno más, por BQSFU la designación de un archivo como un archivo maestro y el UPDATE otros como un archivo de transacciones. En lugar de PROGRAMA simplemente crear un informe, las operaciones se utiliza para actualizar el archivo maestro. Esto procedimiento, conocido como un archivo secuencial actualización, es una aplicación muy común. AUDITORÍA NUEVO INFORME Una lógica similar

se utiliza en otros negocios MASTER solicitudes, así como la fusión / purga la transformación en el marketing directo y en el disco como BQSFUNEW.DAT Informe.txt (será automáticamente la aplicación de los pagos a los créditos en el en el Número de Cliente contabilidad. secuencia.)

El diagrama de flujo del sistema para la secuencia archivo de actualización del programa se muestra a la derecha.

Nota: El propósito de este capítulo es ilustrar la lógica de actualización. "Perfect" de datos se presume, es decir, ningún campo de edición de nivel (por ejemplo, para comprobar que un código postal es numérico) se muestra.

Para mantener nuestro sencillo ejemplo, la entrada (de edad) del archivo maestro, un archivo de transacción, y la producción (nuevo) maestro archivo todos tienen el mismo diseño de registro:

#	Nombre	Descripcion	Inicio	Final	Long	Formato
1	NBR	Numero de Cliente	1	5	5	ZD
2	LNAME	Apellido	6	15	10	CH
3	FNAME	Nombre	16	25	10	CH
4	ADDR	Direccion	26	40	15	CH
5	CITY	Cuidad	41	50	10	CH
6	STATE	Estado	51	52	2	ch
7	ZIP	Codigo Postal	53	57	5	CH
8		*no usado*	58	59	2	
9	ACD	Codigo de Transacion	60	60	1	A/C/D
10	CRLF	Solo para PC/370	61	62	2	CR/LF

Nuestro archivo maestro de entrada, BQSFUMST.DAT, es el siguiente:

```
1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
11224BINFORD   DAN       469 N 400 E    DESOTO    TX75115  A
12111ARIAS     IDA       4028 ELMO LOOP MERCED    CA95340  A
32555RYAN      RICHARD   914 FIFTH ST   NORMAL    IL61761  A
41499HILMER    DEBBIE    21175 FELIPA   BUENA PARKCA90620  A
55123JOSEPHSON PEGGY     248 MICHIGAN   JAMESTOWN NY14701  A
61626HAVLIK    CHERYL    551 WASHINGTON WHITTIER  CA90605  A
77271CARPENTER LOIS      326 BEACH      BERWYN    IL60650  A
81288BLACK     KATHY     618 S ANZA     PASADENA  CA91106  A
81997FOOTE     APRIL     635 BURNS      CAROL STRMIL60187  A
94993DIXSON    RICHARD   1021 BROWN     CHICAGO   IL60612  A
```

El código de transacción en la columna 60 puede ser de tres tipos: A para añade, C para los cambios, o D para eliminaciones. En el archivo maestro (s), el código de transacción indica el tratamiento más recientes aplicados a este registro. Este código es una de todo lo anterior, indicando que no se han producido cambios desde estos registros se han añadido. Nuestro archivo de transacción, BQSFUTRN.DAT, es el siguiente:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678890
12111                    2211 APRICOT   MODESTO   CA95356  C
41499                                                        D
55123AMBROSE   FRANK     220 BARRETT    ROCKFORD  IL61103  A
61627QUALLS    CHERYL    201 N EIGHTH   WHITTIER  CA90605  C
81228                                                        D
82446AMICCI    BRUNO     17397 BARCELON CORVALLIS OR97330  A
```

El campo de clave, número de cliente, es necesario en todas las transacciones. Por agrega, todos los demás campos se siempre también. Para los cambios, sólo los campos a cambiar están llenos pulg Por último, elimina, sólo el número de cliente y código de transacción (D) son necesarios.

Todas las operaciones anteriores son sintácticamente correcta, pero como veremos, no todos son, lógicamente, correcta.

Tanto los archivos de entrada deben estar en secuencia de teclas para las mismas razones, como se discutió en los últimos capítulo. Dada la naturaleza de la transformación, el archivo de salida será automáticamente en la secuencia de teclas también. Empezamos por la lectura de un registro del archivo maestro y un registro de la transacción archivo. Si los campos clave no coinciden, entonces el proceso de registro con el tono más bajo. Por ejemplo, los datos anteriores serán procesadas de la siguiente manera:

```
MASTR TRANS T  NVO
----- ----- - -----
11224 12111 C 11224  Master Baja -
                     Escribir el registro maestro de Nueva Maestro
                     Leer el siguiente registro de grandes maestros.
12111 12111 C 12111  Claves iguales -
                     Cambiar registro existente.
                     Escribir cambiado récord en Nueva Maestro
                     Leer siguiente registro de ambos archivos.
32555 41499 D 32555  Master Baja -
                     Escribir el maestro al Nuevo Maestro
                     Leer el siguiente registro de grandes maestros.
41499 41499 D        Claves iguales -
                     Eliminar registro existente
                     Simplemente no se escribir el registro al nuevo maestro
                     Lee el siguiente registro de ambos archivos.
55123 55123 A 55123  Claves iguales -
                     Agregar un registro existente.
                     Esta es un error
                     Escribir el maestro al Nuevo Maestro
                     Leer el siguiente registro de ambos archivos.
61626 61627 C 61626  Master Baja -
```

```
                    Este es un error de transcripción
                    Escribir el maestro al Nuevo Maestro
                    Leer el siguiente registro de grandes maestros.
77271 61627 C       Transacción bajos -
                    Cambiar un registro que no existe
                    Esto es un error
                    Leer el registro de la transacción siguiente.
77271 81228 D 77271 Master Baja -
                    Escribir el maestro al Nuevo Maestro
                    Leer el siguiente registro de grandes maestros.
81288 81228 D       Transacción bajos -
                    Borrar un registro que no existe
                    Esto es un error
                    Leer el registro de la transacción siguiente.
81288 82446 A 81288 Master Baja -
                    Escribir el maestro al Nuevo Maestro
                    Leer el siguiente registro de grandes maestros.
81997 82446 A 81997 Master Baja -
                    Escribir el maestro al Nuevo Maestro
                    Leer el siguiente registro de grandes maestros.
94993 82446 A 82446 De transacción bajos -
                    Agregar un registro
                    Escribir el registro de transacciones
                    Leer el registro siguiente transacción.
94993 EOF n/a 94993 Archivo de transacciones en EOF -
                    Escribir el registro a la nueva
                    Leer el siguiente registro de grandes maestros.
EOF   EOF n/a EOF   Hecho.
```

La salida del programa es el siguiente. Por supuesto, el informe general, se dará formato, más completamente en lugar de mostrar una tarjeta de la imagen de los registros en que se hizo aquí. Pero nuestro objetivo aquí es demostrar la lógica y la actualización de la salida ha sido intencionalmente simples.

```
A:\MIN>bqsfu
BQSFU ... Begin execution
BQSFU ... Audit list on REPORT.TXT
BQSFU ... Normal end of program
A:\MIN>type report.txt
               Name & Address Update Program                Page   1
                       Audit Listing ----+----1----+----2----+----3----+----4----+----5
12111ARIAS     IDA       4028 ELMO LOOP MERCED    CA95340  A BEFORE CHANGE
12111ARIAS     IDA       2211 APRICOT   MODESTO   CA95356  C AFTER CHANGE
41499HILMER    DEBBIE    21175 FELIPA   BUENA PARKCA90620  A RECORD DELETED
55123JOSEPHSON PEGGY     248 MICHIGAN   JAMESTOWN NY14701  A RECORD ON FILE
55123AMBROSE   FRANK     220 BARRETT    ROCKFORD IL61103   A ADD UNSUCCESSFUL
```

```
61627QUALLS    CHERYL    201 N EIGHTH   WHITTIER  CA90605  C CHNG NOT ON FILE
81228                                                      D DLTE NOT ON FILE
82446AMICCI    BRUNO     17397 BARCELON CORVALLIS OR97330  A ADD SUCCESSFUL
              Name & Address Update Program                 Page   2
                       Audit Listing ----+----1----+----2----+----3----+----4----+----5
Transactions In                   6
Transactions Rejected             3
Old Masters In                   10
Old Masters Deleted               1
Old Masters Changed               1
New Masters Added                 1
New Masters Out                  10
```

La salida (nuevo) del archivo maestro, BQSFUNEW.DAT, es el siguiente:

```
1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
11224BINFORD   DAN       469 N 400 E    DESOTO    TX75115  A
12111ARIAS     IDA       2211 APRICOT   MODESTO   CA95356  C
32555RYAN      RICHARD   914 FIFTH ST   NORMAL    IL61761  A
55123JOSEPHSON PEGGY     248 MICHIGAN   JAMESTOWN NY14701  A
61626HAVLIK    CHERYL    551 WASHINGTON WHITTIER  CA90605  A
77271CARPENTER LOIS      326 BEACH      BERWYN    IL60650  A
81288BLACK     KATHY     618 S ANZA     PASADENA  CA91106  A
81997FOOTE     APRIL     635 BURNS      CAROL STRMIL60187  A
82446AMICCI    BRUNO     17397 BARCELON CORVALLIS OR97330  A
94993DIXSON    RICHARD   1021 BROWN     CHICAGO   IL60612  A
```

El diagrama de flujo del programa, el código ensamblador, y señala a continuación.

12.1. La estructura de la línea principal

BEGIN La estructura de la línea principal de la actualización de el programa es el mismo que vimos en el capítulo anterior: seguimos en SETUP archivos de proceso hasta que el viejo archivo maestro o el archivo de transacción está en EOF. El código BAL siguiente:

280 ⟨*search..., main loop SETUP,EOFMAST,EOFTRANS to EOJ,PROCESS,WRAPUP* 280⟩≡ (289)

```
          BAL    R10,SETUP
 MAIN     EQU    *
          CLI EOFMAST,C'Y'
          BE  EOJ
          CLI EOFTRANS,C'Y'
          BE  EOJ
```

```
          BAL     R10,PROCESS
          B       MAIN
EOJ       EQU     *
          BAL     R10,WRAPUP
```

12.1.1. La rutina de instalación

Dentro de la rutina de instalación, abrimos todos los archivos de y leer el primer registro de la entrada de OPEN archivo maestro y la operación de archivo:

```
SETUP EQU  *
      ST   R10,SVSETUP
      OI   MASTERIN+10,X'08'
      OI   TRANSIN+10,X'08'
      OI   MASTEROT+10,X'08'
      OI   REPORT+10,X'08'
      OPEN MASTERIN
      OPEN TRANSIN
      OPEN MASTEROT
      OPEN REPORT
      BAL  R10,READMST
      BAL  R10,READTRN
      L    R10,SVSETUP
      BR   R10
```

12.2. Las rutinas LEA

LEA

MASTER Tendremos dos rutinas LEA: uno para LEA el archivo maestro de entrada y otro para el MASTER archivo de transacciones. En este programa hemos añadido de la lógica de incluir el número de registros para cada archivo. Estas cuentas se incrementan Y AL FINAL? dentro de las rutinas de LEA. Serán aparece dentro de la rutina Wrapup.

```
READMST  EQU *
         ST  R10,SVREADM
         GET MASTERIN,IREC
         AP  #OLDIN,=P'1'
         B   READMX
ATENDMST EQU *
         MVI EOFMAST,C'Y'
READMX   EQU *
         L   R10,SVREADM
         BR  R10
```

12.2.0.1. La rutina PROCESO

En el momento en que realizamos la rutina PROCESO, Comparación de Número de cliente de archivo principal con el tenemos un registro desde el viejo maestro de archivo Número de Cliente de la y un registro del archivo de transacciones. Nos transacción de archivos. Como en el capítulo anterior, no tres condiciones posibles como resultado de este Comparar:

- TRANS MATCH en el número de cliente en el maestro es menos
- BAJA que el número de clientes en el TRANSACTION, en el número de cliente en el maestro es mayor que el número de clientes en el VOLVER Transacción, o en el número de cliente en el maestro es
- igual al número de clientes en el TRANSACCIÓN.

La primera condición que indique que tener un maestro para los que no hay

```
PROCESS  EQU *
         ST  R10,SVPROC
         CLC INBR,TNBR
         BH  PROC2
         BL  PROC3
         BAL R10,MATCH
         B   PROCESSX
PROC2    EQU *
         BAL R10,TRANSLOW
         B   PROCESSX
PROC3    EQU *
         BAL R10,MASTLOW
PROCESSX EQU *
         L   R10,SVPROC
         BR  R10
```

La segunda condición que indique que tienen una operación para la que no hay maestro. Si el tipo de transacción es un complemento, a continuación, esto es válido. De lo contrario tenemos un intento de Cambiar o eliminar un registro que no

La tercera condición que indique que tienen una operación por un maestro que ya existen. Si el tipo de transacción es cambiar o borrar, a continuación, esto es válido. De lo contrario, tenemos un intento de Agregar un registro que ya existe. Cada una de estas condiciones serán tratados en un de rutina por separado.

12.2.0.2. Master Low

Low MUDANZA DE EDAD El "Maestro de baja condición" indica que tenemos un CAMPOS MASTER A LOS NUEVOS registro maestro sin realizar

una transacción correspondiente. Esto no es el amo un error: acaba de escribir este registro para el nuevo archivo maestro y el leer el siguiente registro del archivo maestro de edad.

```
MASTLOW EQU *
        ST  R10,SVMSTLOW
        MVC OREC,IREC
        BAL R10,WRITENEW
        BAL R10,READMST
        L   R10,SVMSTLOW
        BR  R10
```

El "Trans baja condición" (también conocido como "Alto Maestro") indica que tenemos una transacción sin coincidentes registro del archivo maestro. Sea o no esto es un error depende del tipo de transacción. Si es un complemento, entonces es una operación válida. Si es un cambiar o borrar, entonces es un error: nos no se puede cambiar o borrar lo que no está allí.

12.2.1. transacción bajos

```
TRANSLOW EQU *
         ST  R10,SVTRNLOW
         CLI TACD,ADD
         BE  TRANSLO2
         CLI TACD,CHANGE
         BE  TRANSLO3
         CLI TACD,DELETE
         BE  TRANSLO4
         BAL R10,BADCODE
         BAL R10,READTRN
         B   TRANSLOX
TRANSLO2 EQU *
         BAL R10,ADDIT
         BAL R10,READTRN
         B   TRANSLOX
TRANSLO3 EQU *
         BAL R10,BADCHANG
         BAL R10,READTRN
         B   TRANSLOX
TRANSLO4 EQU *
         BAL R10,BADDELET
         BAL R10,READTRN
TRANSLOX EQU *
         L   R10,SVTRNLOW
         BR  R10
```

12.2.1.1. Válido Añadir

complemento válido Una operación ADD con Trans baja es válida ...

```
ADDIT  EQU *
       ST  R10,SVADDIT
       MVC ONBR,TNBR
       MVC OLNAME,TLNAME
       MVC OFNAME,TFNAME
       MVC OADDR,TADDR
       MVC OCITY,TCITY
       MVC OSTATE,TSTATE
       MVC OZIP,TZIP
       MVC OACD,TACD
       MVC OCRLF,TCRLF
       BAL R10,WRITENEW
       BAL R10,CHKLNS
       MVC RREC,BLANKS
       BAL R10,WRITE
       MVC RDATA,TREC
       MVC RMSG,=CL16'ADD SUCCESSFUL'
       BAL R10,WRITE
       AP  #ADDED,=P'1'
ADDITX EQU *
       L   R10,SVADDIT
       BR  R10
```

12.2.2. No válido el cambio, no se puede eliminar

AGREGADOS Una transacción de cambio o DELETE con Trans de bajo es el rendimiento de un error ...

```
BADCHANG EQU *
         ST  R10,SVBADCHG
         BAL R10,CHKLNS
         MVC RREC,BLANKS
         BAL R10,WRITE
         MVC RDATA,TREC
         MVC RMSG,=CL16'CHNG NOT ON FILE'
         BAL R10,WRITE
         AP  #REJECTS,=P'1'
BADCHGX EQU  *
         L   R10,SVBADCHG
         BR  R10
BADDELET EQU *
         ST  R10,SVBADDEL
         BAL R10,CHKLNS
```

```
         MVC RREC,BLANKS
         BAL R10,WRITE
         MVC RDATA,TREC
         MVC RMSG,=CL16'DLTE NOT ON FILE'
         BAL R10,WRITE
         AP  #REJECTS,=P'1'
BADDELX EQU  *
         L   R10,SVBADDEL
         BR  R10
```

12.2.2.1. Bad Código de transacción

BAD CODE En el capítulo anterior utiliza dos archivos de entrada, sino que se mueven TRANS Campos para allí no estaban interesados en la transacción AUDITORÍA códigos. En este programa, el código de transacción debe ser A, C o D. Todo lo demás es una condición de error ...

```
BADCODE  EQU *
         ST  R10,SVBADCOD
         BAL R10,CHKLNS
         MVC RREC,BLANKS
         BAL R10,WRITE
         MVC RDATA,TREC
         MVC RMSG,=CL16'CODE NOT A/C/D'
         BAL R10,WRITE
         AP  #REJECTS,=P'1'
BADCODEX EQU *
         L   R10,SVBADCOD
         BR  R10
```

12.2.2.2. Master / transacción del partido

El "Match" condición indica que tenemos una operación con un registro maestro de juego. Una vez más, si es o no es un error depende del tipo de transacción. Si se trata de un TDA, entonces es un error: no se puede agregar un registro con la misma clave, como un registro existente. Si es un cambio o DELETE, entonces es válido: estamos tratando de cambiar o eliminar un registro existente.

```
MATCH EQU *
       ST  R10,SVMATCH
       CLI TACD,ADD
       BE  MATCH2
       CLI TACD,CHANGE
       BE  MATCH3
       CLI TACD,DELETE
       BE  MATCH4
```

```
        BAL R10,BADCODE
        BAL R10,READTRN
        B   MATCHX
MATCH2  EQU *
        BAL R10,BADADD
        BAL R10,READTRN
        B   MATCHX
MATCH3  EQU *
        BAL R10,CHANGEIT
        BAL R10,READTRN
        BAL R10,READMST
        B   MATCHX
MATCH4  EQU *
        BAL R10,DELETEIT
        BAL R10,READTRN
        BAL R10,READMST
MATCHX  EQU *
        L   R10,SVMATCH
        BR  R10
```

12.2.3. Añadir malo

añade Una operación ADD con un registro maestro coincidentes siempre es válido ...

```
BADADD  EQU *
        ST  R10,SVBADADD
        BAL R10,CHKLNS
        MVC RREC,BLANKS
        BAL R10,WRITE
        MVC RDATA,IREC
        MVC RMSG,=CL16'RECORD ON FILE'
        BAL R10,WRITE
        MVC RDATA,TREC
        MVC RMSG,=CL16'ADD UNSUCCESSFUL'
        BAL R10,WRITE
        AP  #REJECTS,=P'1'
BADADDX EQU *
        L   R10,SVBADADD
        BR  R10
```

12.2.4. Válido Eliminar

Una operación DELETE con un maestro coincidentes CAMPOS DE MASTER A registro de auditoría es válida. Para borrar un registro de la antigua Archivo maestro simplemente no escribir a la nueva Archivo maestro ...

```
DELETEIT EQU *
         ST  R10,SVDELIT
         BAL R10,CHKLNS
         MVC RREC,BLANKS
         BAL R10,WRITE
         MVC RDATA,IREC
         MVC RMSG,=CL16'RECORD DELETED'
         BAL R10,WRITE
         AP  #DELETED,=P'1'
DELETEX  EQU  *
         L   R10,SVDELIT
         BR  R10
```

12.2.5. Válido Cambio

Una transacción de cambio con un registro maestro juego es válido. Copia del registro maestro de edad a la nuevo registro maestro. Mueva todos los campos no transacción en blanco en los campos correspondientes en el nuevo Registro maestro. Escriba el nuevo registro maestro. Mostrar "antes" y "después" de las imágenes en la auditoría informe.

```
CHANGEIT EQU   *
         ST    R10,SVCHGIT
*
* COPY EXISTING RECORD TO OUTPUT RECORD
* THEN MAKE THE REQUESTED CHANGES
*
         MVC   OREC,IREC
         CLC   TLNAME,BLANKS
         BE    CHGIT2
         MVC   OLNAME,TLNAME
CHGIT2   EQU   *
         CLC   TFNAME,BLANKS
         BE    CHGIT3
         MVC   OFNAME,TFNAME
CHGIT3   EQU   *
         CLC   TADDR,BLANKS
         BE    CHGIT4
         MVC   OADDR,TADDR
CHGIT4   EQU   *
         CLC   TCITY,BLANKS
         BE    CHGIT5
         MVC   OCITY,TCITY
CHGIT5   EQU   *
         CLC   TSTATE,BLANKS
         BE    CHGIT6
```

```
         MVC   OSTATE,TSTATE
CHGIT6   EQU   *
         CLC   TZIP,BLANKS
         BE    CHGIT7
         MVC   OZIP,TZIP
CHGIT7   EQU   *
         MVC   OACD,TACD
         BAL   R10,WRITENEW
*
* SHOW RECORD BEFORE AND AFTER CHANGES
*
         BAL   R10,CHKLNS
         MVC   RREC,BLANKS
         BAL   R10,WRITE
         MVC   RDATA,IREC
         MVC   RMSG,=CL16'BEFORE CHANGE'
         BAL   R10,WRITE
         MVC   RDATA,OREC
         MVC   RMSG,=CL16'AFTER CHANGE'
         BAL   R10,WRITE
         AP    #CHANGED,=P'1'
CHGITX   EQU   *
         L     R10,SVCHGIT
         BR    R10
```

12.2.6. Escribir nuevo Master

nuevo MASTER Recordemos que el nuevo archivo maestro es independiente de la ESCRIBIR NUEVO el archivo maestro de edad. No hay nada inusual en el Master esta rutina. Se incluye un recuento de los registros de por escrito.

```
WRITENEW EQU *
         ST  R10,SVWRITEN
         PUT MASTEROT,OREC
         AP  #NEWOUT,=P'1'
         L   R10,SVWRITEN
         BR  R10
```

12.2.7. La rutina Wrapup

La rutina de Wrapup para este programa es muy similar a la mostrada en el capítulo anterior. Recordemos que seguimos el proceso hasta que los registros del archivo maestro o el archivo de transacción está en EOF. Por lo tanto, cuando lleguemos a Wrapup, es probable que uno de estos archivos no está en EOF. Nosotros por lo tanto ejecutar uno de los siguientes ciclos:

- Si el archivo maestro de edad no está en EOF, entonces el proceso todos los registros de archivo que queda MASTER como sin precedentes, es decir, bajo maestro. (Recuerde estos registros se escriben en el MASTER nuevo archivo de sin cambios.)
- De lo contrario, si el archivo de transacción no está en EOF, entonces Wrapup procesamos todos los registros de los archivos restantes de transacción como sin precedentes, es decir, baja de transacción. (Recuérdese MASTER AGREGA incomparable será añadido, mientras que incomparable son errores.)

```
WRAPUP  EQU   *
        ST    R10,SVWRAP
WRAPUP2 EQU   *
        CLI   EOFMAST,C'Y'
        BE    WRAPUP3
        BAL   R10,MASTLOW
        B     WRAPUP2
WRAPUP3 EQU   *
        CLI   EOFTRANS,C'Y'
        BE    WRAPUP4
        BAL   R10,TRANSLOW
        B     WRAPUP3
WRAPUP4 EQU   *
        CLOSE MASTERIN
        CLOSE TRANSIN
        CLOSE MASTEROT
        BAL   R10,DOCOUNTS
        CLOSE REPORT
        WTO   'BQSFU ... Audit list on REPORT.TXT'
        L     R10,SVWRAP
        BR    R10
```

12.2.8. Programa de Solución

El programa completa y comentada, BQSFU.MLC, sigue.

289 ⟨ff 40⟩+≡ ◁249 385▷

```
        PRINT NOGEN
*****************************************************************
*       FILENAME: BQSFU.MLC                                     *
*       AUTHOR : Bill Qualls                                    *
*       SYSTEM : PC/370 R4.2                                    *
*       REMARKS : Sequential File Update Sample Program         *
*****************************************************************
        START 0
        REGS
```

```
BEGIN    BEGIN
         WTO    'BQSFU ... Begin execution'
⟨search..., main loop SETUP,EOFMAST,EOFTRANS to EOJ,PROCESS,WRAPUP 280⟩
         WTO    'BQSFU ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,                *
*                 before any records are processed.                      *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     MASTERIN+10,X'08' PC/370 ONLY - Convert all
*                                        input from ASCII to EBCDIC
         OI     TRANSIN+10,X'08'          PC/370 ONLY - Convert all
*                                        input from ASCII to EBCDIC
         OI     MASTEROT+10,X'08' PC/370 ONLY - Convert all
*                                        output from EBCDIC to ASCII
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                        output from EBCDIC to ASCII
         OPEN MASTERIN
         OPEN TRANSIN
         OPEN MASTEROT
         OPEN REPORT
         BAL    R10,READMST
         BAL    R10,READTRN
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        HDGS - Print headings.                                          *
*****************************************************************
HDGS     EQU    *
         ST     R10,SVHDGS
         AP     PGS,=P'1'                 Add 1 to page count
         MVC    HDPGS,=X'40202120' Edit pattern for page count
         ED     HDPGS,PGS                 Move page count to heading
         PUT    REPORT,FORMFEED           PC/370 ONLY
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         ZAP   LNS,=P'0'                  Reset line count to zero
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.  *
*****************************************************************
```

```
PROCESS EQU    *
         ST    R10,SVPROC
         CLC   INBR,TNBR                  Attempt match on customer nbr
         BH    PROC2                      Transaction low
         BL    PROC3                      Master low
         BAL   R10,MATCH                  Otherwise a match was found
         B     PROCESSX
PROC2    EQU   *                          No master for this transaction
         BAL   R10,TRANSLOW
         B     PROCESSX
PROC3    EQU   *                          No transaction for this master
         BAL   R10,MASTLOW
PROCESSX EQU   *
         L     R10,SVPROC
         BR    R10
*************************************************************
*        MASTLOW - No updates for this master record.                *
*                  Just write this record and go to next.            *
*************************************************************
MASTLOW EQU    *
         ST    R10,SVMSTLOW
         MVC   OREC,IREC                  Move input to output
         BAL   R10,WRITENEW               Write new master record
         BAL   R10,READMST                Read next master record
         L     R10,SVMSTLOW
         BR    R10
*************************************************************
*        MATCH - Transaction for existing master record.             *
*************************************************************
MATCH    EQU   *
         ST    R10,SVMATCH
         CLI   TACD,ADD                   Attempt to add?
         BE    MATCH2                     Can't do it - already exists
         CLI   TACD,CHANGE                Attempt to change?
         BE    MATCH3                     OK to change existing record
         CLI   TACD,DELETE                Attempt to delete?
         BE    MATCH4                     OK to delete existing record
         BAL   R10,BADCODE                Error - unrecognized code
         BAL   R10,READTRN                Read next transaction
         B     MATCHX
MATCH2   EQU   *                          Attempt to add
         BAL   R10,BADADD                 Can't add - it already exists
         BAL   R10,READTRN                Read next transaction
         B     MATCHX
MATCH3   EQU   *                          Attempt to change
         BAL   R10,CHANGEIT               OK to change existing record
```

```
         BAL   R10,READTRN              Read next transaction
         BAL   R10,READMST              Read next master
         B     MATCHX
MATCH4   EQU   *                        Attempt to delete
         BAL   R10,DELETEIT             OK to delete existing record
         BAL   R10,READTRN             Read next transaction
         BAL   R10,READMST             Read next master
MATCHX   EQU   *
         L     R10,SVMATCH
         BR    R10
*****************************************************************
*        TRANSLOW - Transaction without a matching master.                *
*****************************************************************
TRANSLOW EQU   *
         ST    R10,SVTRNLOW
         CLI   TACD,ADD                 Attempt to add?
         BE    TRANSLO2                 OK since it doesn't exist
         CLI   TACD,CHANGE              Attempt to change?
         BE    TRANSLO3                 Can't change - doesn't exist
         CLI   TACD,DELETE              Attempt to delete?
         BE    TRANSLO4                 Can't delete - not there
         BAL   R10,BADCODE              Error - Unrecognized code
         BAL   R10,READTRN              Read next transaction
         B     TRANSLOX
TRANSLO2 EQU   *                        Attempt to add
         BAL   R10,ADDIT                OK to add - not there already
         BAL   R10,READTRN              Read next transaction
         B     TRANSLOX
TRANSLO3 EQU   *                        Attempt to change
         BAL   R10,BADCHANG             Can't change - doesn't exist
         BAL   R10,READTRN              Read next transaction
         B     TRANSLOX
TRANSLO4 EQU   *                        Attempt to delete
         BAL   R10,BADDELET             Can't delete - doesn't exist
         BAL   R10,READTRN              Read next transaction
TRANSLOX EQU   *
         L     R10,SVTRNLOW
         BR    R10
*****************************************************************
*        BADCODE - Bad Transaction Code                                   *
*****************************************************************
BADCODE EQU    *
         ST    R10,SVBADCOD
         BAL   R10,CHKLNS
         MVC   RREC,BLANKS
         BAL   R10,WRITE
```

```
         MVC   RDATA,TREC
         MVC   RMSG,=CL16'CODE NOT A/C/D'
         BAL   R10,WRITE
         AP    #REJECTS,=P'1'
BADCODEX EQU   *
         L     R10,SVBADCOD
         BR    R10
*****************************************************************
*        BADADD - Bad Add Attempted                             *
*****************************************************************
BADADD   EQU   *
         ST    R10,SVBADADD
         BAL   R10,CHKLNS
         MVC   RREC,BLANKS
         BAL   R10,WRITE
         MVC   RDATA,IREC
         MVC   RMSG,=CL16'RECORD ON FILE'
         BAL   R10,WRITE
         MVC   RDATA,TREC
         MVC   RMSG,=CL16'ADD UNSUCCESSFUL'
         BAL   R10,WRITE
         AP    #REJECTS,=P'1'
BADADDX EQU    *
         L     R10,SVBADADD
         BR    R10
*****************************************************************
*        BADCHG - Bad Change Attempted                          *
*****************************************************************
BADCHANG EQU   *
         ST    R10,SVBADCHG
         BAL   R10,CHKLNS
         MVC   RREC,BLANKS
         BAL   R10,WRITE
         MVC   RDATA,TREC
         MVC   RMSG,=CL16'CHNG NOT ON FILE'
         BAL   R10,WRITE
         AP    #REJECTS,=P'1'
BADCHGX EQU    *
         L     R10,SVBADCHG
         BR    R10
*****************************************************************
*        BADDEL - Bad Delete Attempted                          *
*****************************************************************
BADDELET EQU   *
         ST    R10,SVBADDEL
         BAL   R10,CHKLNS
```

```
         MVC   RREC,BLANKS
         BAL   R10,WRITE
         MVC   RDATA,TREC
         MVC   RMSG,=CL16'DLTE NOT ON FILE'
         BAL   R10,WRITE
         AP    #REJECTS,=P'1'
BADDELX  EQU   *
         L     R10,SVBADDEL
         BR    R10
*****************************************************************
*        ADDIT - Add a new record to master file                *
*****************************************************************
ADDIT    EQU   *
         ST    R10,SVADDIT
         MVC   ONBR,TNBR
         MVC   OLNAME,TLNAME
         MVC   OFNAME,TFNAME
         MVC   OADDR,TADDR
         MVC   OCITY,TCITY
         MVC   OSTATE,TSTATE
         MVC   OZIP,TZIP
         MVC   OACD,TACD
         MVC   OCRLF,TCRLF
         MVC   OREC,TREC
         BAL   R10,WRITENEW
         BAL   R10,CHKLNS
         MVC   RREC,BLANKS
         BAL   R10,WRITE
         MVC   RDATA,TREC
         MVC   RMSG,=CL16'ADD SUCCESSFUL'
         BAL   R10,WRITE
         AP    #ADDED,=P'1'
ADDITX   EQU   *
         L     R10,SVADDIT
         BR    R10
*****************************************************************
*        CHANGEIT - Apply changes to existing master record     *
*****************************************************************
CHANGEIT EQU   *
         ST    R10,SVCHGIT
*
*        COPY EXISTING RECORD TO OUTPUT RECORD
*        THEN MAKE THE REQUESTED CHANGES
*
         MVC   OREC,IREC
         CLC   TLNAME,BLANKS
```

```
         BE    CHGIT2
         MVC   OLNAME,TLNAME
CHGIT2   EQU   *
         CLC   TFNAME,BLANKS
         BE    CHGIT3
         MVC   OFNAME,TFNAME
CHGIT3   EQU   *
         CLC   TADDR,BLANKS
         BE    CHGIT4
         MVC   OADDR,TADDR
CHGIT4   EQU   *
         CLC   TCITY,BLANKS
         BE    CHGIT5
         MVC   OCITY,TCITY
CHGIT5   EQU   *
         CLC   TSTATE,BLANKS
         BE    CHGIT6
         MVC   OSTATE,TSTATE
CHGIT6   EQU   *
         CLC   TZIP,BLANKS
         BE    CHGIT7
         MVC   OZIP,TZIP
CHGIT7   EQU   *
         MVC   OACD,TACD
         BAL   R10,WRITENEW
*
*        SHOW THE RECORD BEFORE AND AFTER CHANGES
*
         BAL   R10,CHKLNS
         MVC   RREC,BLANKS
         BAL   R10,WRITE
         MVC   RDATA,IREC
         MVC   RMSG,=CL16'BEFORE CHANGE'
         BAL   R10,WRITE
         MVC   RDATA,OREC
         MVC   RMSG,=CL16'AFTER CHANGE'
         BAL   R10,WRITE
         AP    #CHANGED,=P'1'
CHGITX   EQU   *
         L     R10,SVCHGIT
         BR    R10
*************************************************************
*        DELETEIT - Delete an existing master record                *
*                   (To delete it, just don't write it out.)        *
*************************************************************
DELETEIT EQU   *
```

```
                ST    R10,SVDELIT
                BAL   R10,CHKLNS
                MVC   RREC,BLANKS
                BAL   R10,WRITE
                MVC   RDATA,IREC
                MVC   RMSG,=CL16'RECORD DELETED'
                BAL   R10,WRITE
                AP    #DELETED,=P'1'
        DELETEX EQU   *
                L     R10,SVDELIT
                BR    R10
        ************************************************************
        *       READMST - Read a master record.                    *
        ************************************************************
        READMST EQU   *
                ST    R10,SVREADM
                GET   MASTERIN,IREC
                AP    #OLDIN,=P'1'
                B     READMX
        ATENDMST EQU  *
                MVI   EOFMAST,C'Y'
        READMX  EQU   *
                L     R10,SVREADM
                BR    R10
        ************************************************************
        *       READOFF - Read a transaction record.               *
        ************************************************************
        READTRN EQU   *
                ST    R10,SVREADT
                GET   TRANSIN,TREC
                AP    #TRANSIN,=P'1'
                B     READTX
        ATENDTRN EQU  *
                MVI   EOFTRANS,C'Y'
        READTX  EQU   *
                L     R10,SVREADT
                BR    R10
***************************************************************
*       CHKLNS - Check lines printed. Full page?              *
***************************************************************
⟨search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS 184⟩
        ************************************************************
        *       WRITE - Write a single detail line.                *
        ************************************************************
        WRITE   EQU   *
                ST    R10,SVWRITE
```

```
         PUT   REPORT,RREC              Write report line
         AP    LNS,=P'1'
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRITE - Write a new master record.                     *
*****************************************************************
WRITENEW EQU   *
         ST    R10,SVWRITEN
         PUT   MASTEROT,OREC
         AP    #NEWOUT,=P'1'
         L     R10,SVWRITEN
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
*                                       At this point we know that
*                                       at least one of the input
*                                       files is at EOF. Process
*                                       other file as "unmatched"
*                                       until at EOF also.
WRAPUP2 EQU    *
         CLI   EOFMAST,C'Y'
         BE    WRAPUP3
         BAL   R10,MASTLOW
         B     WRAPUP2
WRAPUP3 EQU    *
         CLI   EOFTRANS,C'Y'
         BE    WRAPUP4
         BAL   R10,TRANSLOW
         B     WRAPUP3
WRAPUP4 EQU    *
         CLOSE MASTERIN
         CLOSE TRANSIN
         CLOSE MASTEROT
         BAL   R10,DOCOUNTS
         CLOSE REPORT
         WTO   'BQSFU ... Audit list on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        DOCOUNTS - Show counts for audit                       *
*****************************************************************
```

```
DOCOUNTS EQU   *
         ST    R10,SVCOUNTS
         BAL   R10,HDGS
         MVC   AREC,BLANKS
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'Transactions In'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#TRANSIN
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'Transactions Rejected'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#REJECTS
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'Old Masters In'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#OLDIN
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'Old Masters Deleted'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#DELETED
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'Old Masters Changed'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#CHANGED
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'New Masters Added'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#ADDED
         BAL   R10,WRITE
*
         MVC   ADESC,=CL25'New Masters Out'
         MVC   ACOUNT,EDCOUNT
         ED    ACOUNT,#NEWOUT
         BAL   R10,WRITE
*
         L     R10,SVCOUNTS
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
```

```
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
MASTERIN DCB   LRECL=62,RECFM=F,MACRF=G,EODAD=ATENDMST,
               DDNAME='BQSFUMST.DAT'
TRANSIN DCB    LRECL=62,RECFM=F,MACRF=G,EODAD=ATENDTRN,
               DDNAME='BQSFUTRN.DAT'
MASTEROT DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='BQSFUNEW.DAT'
REPORT   DCB   LRECL=80,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP DC     F'0'                    SETUP
SVHDGS   DC    F'0'                    HDGS
SVPROC   DC    F'0'                    PROCESS
SVREADM DC     F'0'                    READMST
SVREADT DC     F'0'                    READTRN
SVWRITE DC     F'0'                    WRITE
SVWRITEN DC    F'0'                    WRITENEW
SVWRAP   DC    F'0'                    WRAPUP
SVCHKLNS DC    F'0'                    CHKLNS
SVMATCH DC     F'0'                    MATCH
SVMSTLOW DC    F'0'                    MASTLOW
SVTRNLOW DC    F'0'                    TRANSLOW
SVCOUNTS DC    F'0'                    DOCOUNTS
SVBADCOD DC    F'0'                    BADCODE
SVBADADD DC    F'0'                    BADADD
SVBADCHG DC    F'0'                    BADCHANG
SVBADDEL DC    F'0'                    BADDELET
SVADDIT DC     F'0'                    ADDIT
SVCHGIT DC     F'0'                    CHANGEIT
SVDELIT DC     F'0'                    DEELTEIT
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFMAST  DC    CL1'N'                   End of master file? (Y/N)
EOFTRANS DC    CL1'N'                   End of transaction file? (Y/N)
EDCOUNT  DC    X'40206B2020206B202120'             BZ,ZZZ,ZZ9
PGS      DC    PL2'0'                   Nbr of pages printed.
LNS      DC    PL2'20'                  Lines printed on this page.
MAXLNS   DC    PL2'20'                  Max nbr lines per page.
*                                       My line counts exclude hdgs.
BLANKS   DS    0CL80
```

```
         DC    CL78' ',XL2'0D25'
*****************************************************************
*        Transaction codes                                      *
*****************************************************************
ADD      EQU   C'A'
CHANGE   EQU   C'C'
DELETE   EQU   C'D'
*****************************************************************
*        Counts for audit purposes                              *
*****************************************************************
#TRANSIN DC    PL4'0'       Transactions In
#REJECTS DC    PL4'0'       Transactions Rejected
#OLDIN   DC    PL4'0'       Old Masters In
#DELETED DC    PL4'0'       Old Masters Deleted
#CHANGED DC    PL4'0'       Old Masters Changed
#ADDED   DC    PL4'0'       New Masters Added
#NEWOUT DC     PL4'0'       New Masters Out
*****************************************************************
*        Input record definition - Master In                    *
*****************************************************************
IREC     DS    0CL62         1-62         Master record
INBR     DS    CL5           1- 5         Customer nbr
ILNAME   DS    CL10          6-15         Last name
IFNAME   DS    CL10         16-25         First name
IADDR    DS    CL15         26-40         Address
ICITY    DS    CL10         41-50         City
ISTATE   DS    CL2          51-52         State
IZIP     DS    CL5          53-57         Zip
         DS    CL2          58-59         Unused
IACD     DS    CL1          60-60         Transaction code (A/C/D)
ICRLF    DS    CL2          61-62         PC/370 only - CR/LF
*****************************************************************
*        Input record definition - Transaction                  *
*****************************************************************
TREC     DS    0CL62         1-62         Transaction record
TNBR     DS    CL5           1- 5         Customer nbr
TLNAME   DS    CL10          6-15         Last name
TFNAME   DS    CL10         16-25         First name
TADDR    DS    CL15         26-40         Address
TCITY    DS    CL10         41-50         City
TSTATE   DS    CL2          51-52         State
TZIP     DS    CL5          53-57         Zip
         DS    CL2          58-59         Unused
TACD     DS    CL1          60-60         Transaction code (A/C/D)
TCRLF    DS    CL2          61-62         PC/370 only - CR/LF
*****************************************************************
```

```
*        Output record definition - Master Out                  *
*****************************************************************
OREC     DS    0CL62         1-62         Master record
ONBR     DS    CL5           1- 5         Customer nbr
OLNAME   DS    CL10          6-15         Last name
OFNAME   DS    CL10         16-25         First name
OADDR    DS    CL15      26-40    Address
OCITY    DS    CL10      41-50    City
OSTATE   DS    CL2       51-52    State
OZIP     DS    CL5       53-57    Zip
         DS    CL2       58-59    Unused
OACD     DS    CL1       60-60    Transaction code (A/C/D)
OCRLF    DS    CL2       61-62    PC/370 only - CR/LF
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
RREC     DS    0CL80      1-80    Report record
RDATA    DC    CL60' '    1-60    Transaction Data
         DC    CL2' '    61-62
RMSG     DC    CL16' '   63-78    Audit message
RCRLF    DS    CL2       79-80    PC/370 only - CR/LF
*****************************************************************
*        Output record definition - Audit                       *
*        !!! NOTE HOW SPACE FOR RREC IS REDEFINED !!!           *
*****************************************************************
         ORG   RREC
AREC     DS    0CL80      1-87    Audit Line
ADESC    DC    CL25' '    1-25    Description on count
ACOUNT   DC    CL10' '   26-35    Count
         DC    CL43' '   36-78
ACRLF    DS    CL2       79-80    PC/370 only - CR/LF
         ORG
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
FORMFEED DS    0CL80              PC/370 only
*        DC    X'0C'              EBCDIC formfeed
*        DC    CL77' '
         DC    78C'_'             For testing...
         DC    X'0D25'            EBCDIC CR/LF
HD1      DS    0CL80
         DC    CL40'               Name & Address Update Pro'
         DC    CL26'gram                  Page'
HDPGS    DC    CL4'BZZ9'
         DC    CL8' '
         DC    XL2'0D25'
```

```
HD2      DS    0CL80
         DC    CL78'                          Audit Listing'
         DC    XL2'0D25'
HD3      DS    0CL80
         DC    CL78' '
         DC    XL2'0D25'
HD4      DS    0CL80
         DC    CL40'--+--1--+--2--+--3--+--4'
         DC    CL38'--+--5--+--6 MESSAGES'
         DC    XL2'0D25'
         END   BEGIN
```

Observe el uso de la instrucción ORG arriba. Esto nos permite "definir" un registro. La primera la instrucción, ORG RREC, dice que para restablecer el contador de localización (recuerde: "cosas de la izquierda") a lo que Fue en RREC arriba. La segunda instrucción, ORG (solo), dice que para poner el contador de posición de nuevo a lo que era antes de la primera ORG. Mediante el uso de la ORG, hemos ahorrado 80 bytes de memoria. Cuándo utilizando ORG, es fácil olvidar el "cierre" ORG. No lo hagas!

12.2.9. Conciliar el informe de auditoría

Echemos una mirada más en el informe de auditoría elaborado por este programa:

```
Name & Address Update Program                Page   2
                       Audit Listing
----+----1----+----2----+----3----+----4----+----5----+----6 MESSAGES
Transactions In                   6
Transactions Rejected             3
Old Masters In                   10
Old Masters Deleted               1
Old Masters Changed               1
New Masters Added                 1
New Masters Out                  10
```

Es importante que seamos capaces de conciliar la cuenta en este informe, es decir, tenemos que ser capaces de cuenta de todas las transacciones y los registros maestros. Por ejemplo, vemos que hay seis Entonces, ¿qué operaciones pulgadas lo que les sucedió? Tres fueron rechazadas por errores, uno fue un válido eliminar, uno era un cambio válido, y uno era un complemento válido: $3 + 1 + 1 + 1 = 6$.

Vemos que había 10 grandes maestros y 10 maestros nuevos a cabo. ¿Cómo conciliar estas cuenta? Vemos que un registro se ha eliminado, por lo que nos lleva de diez a nueve. Un registro se ha cambiado, pero los cambios no tienen efecto en el número de registros. Por último, un registro se añadido, por lo que nos lleva de nueve hoyos hasta diez: $10 - 1 + 1 = 10$.

12.3. Ejercicios

1. Verdadero o falso. Cuando dos archivos se procesan utilizando lógica secuencial de archivo de actualización ...

   a) Hay tres tipos de operaciones. F T

   b) ambos archivos deben estar en "clave" de la secuencia. F T

   c) de la lógica de largo recorrido continuará hasta que ambos archivos se encuentran en EOF. F T

   d) la rutina de instalación contendrá un oscurecimiento de lectura para ambos archivos. F T

   e) una operación de ADD en Match es un error. F T

   f) una operación de CAMBIO en Match es un error. F T

   g) una operación DELETE en Match es un error. F T

   h) una operación de ADD en TRANSLOW es un error. F T

   i) una transacción CAMBIO en TRANSLOW es un error. F T

   j) una operación DELETE en TRANSLOW es un error. F T

   k) leemos la próxima MASTER y registros de transacciones al final de la PROCESO F T de rutina.

   l) Nuevos amos = Old Masters + agrega válido - Borra válida. F T

   m) OPERACIONES EN LAS OPERACIONES = rechazó + agrega válido - Borra válido.

2. La siguiente tabla muestra el número de registros en el archivo el viejo maestro, y la clave y la tipo de transacción para los registros en el archivo de transacciones. Determinar qué registros se escrito a presentar la nueva obra maestra, y el suministro de la cuenta para el informe de auditoría.

old Master	Trans Key	Type	new Master
112	222	d	
222	223	a	
317	496	a	
469	558	d	
558	628	d	
627	731	c	
731	808	d	
880	914	a	
914			
921			

   Informe de auditoría

En las transacciones	
Transacciones rechazadas	
En los viejos maestros	
Old Masters eliminados	
Old Masters cambiado	
Nueva Añadido	
Masters Nuevo	
máster de salida	

3. ¿Cuál de los siguientes informes de auditoría indican un error de programa?

```
(a) Transactions In       35 (b) Transactions In       23
    Transactions Rejected  5     Transactions Rejected  3
    Old Masters In        50     Old Masters In        10
    Old Masters Deleted   10     Old Masters Deleted    6
    Old Masters Changed    5     Old Masters Changed    7
    New Masters Added     15     New Masters Added      8
    New Masters Out       60     New Masters Out       12
```

4. En el programa de ejemplo, BQSFU.MLC, asumimos un campo en blanco en una transacción de cambio de indicó que el campo correspondiente en el registro maestro no debe ser cambiado. Pero ¿Cómo nos indican que un campo debe ser cambiado a los espacios en blanco? Por ejemplo, podemos sabemos que un cliente se ha mudado a una nueva ciudad, pero no sabemos el nuevo código postal. Uno solución común a este tipo de problemas es la designación de algunos caracteres especiales (tales como signo de dólar) que, si se encuentra en la primera posición de un campo en el registro de transacciones, indica que los espacios en blanco deben ser trasladados al campo correspondiente en el registro maestro. Cambiar el programa de la muestra para incluir esta característica. Prueba con las siguientes operaciones:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678
                                                                (changed)
12111                    2211 APRICOT   MODESTO   CA$       C
41499                                                         D
55123AMBROSE   FRANK     220 BARRETT    ROCKFORD  IL61103   A
61627QUALLS    CHERYL    201 N EIGHTH   WHITTIER  CA90605   C
81228                                                         D
                                                                (new)
81997          KEVEN     $              WEST CHGO IL60185   C
82446AMICCI    BRUNO     17397 BARCELON CORVALLIS OR97330   A
```

5. In the sample program, BQSFU.MLC , deleted records were physically deleted; that is, those 5. records were no longer included in the MASTER file. Sometimes we prefer records be logically deleted; that is, the records are still included in the MASTER file, but they are tagged in such a way as to indicate that they should not be processed by other programs. (This is how dBASE handles deleted records: an asterisk in the first position of the record is used to indicate logically deleted records. dBASE's PACK command is used to physically delete them.) We have included the most recent transaction code in position 60 of the master record. We could say a D in this position indicates the record is logically deleted.

   a) Change the sample program to include this feature.

   b) Show how the READ routine would be modified for all other programs using this file.

   c) Write a program which would physically delete all records which are currently logically deleted (similar to dBASE's PACK).

   d) Write a program which would "undelete" all records which are currently logically deleted (similar to dBASE's RECALL).

6. (Refer to the Small Town Payroll database in More Datasets.) The following table describes the transactions used to update the EMPL file:

Field Nbr	Name	Field Description	Begins	Ends	Len	Format
1	NUM	Employee number	1	3	3	ZD
2	LNAME	Last name	4	13	10	CH
3	FNAME	First name	14	23	10	CH
4	DEPT	Department	24	24	1	CH
5	RATE	Pay rate	25	29	5	999V99
6	TYPE	Rate type	30	30	1	H or S
7	ACD	Transaction code	31	31	1	A/C/D
8	CRLF	PC/370 Only	32	33	2	CR/LF

   Escriba un programa que se aplicará a las operaciones siguientes en el archivo EMPL.

```
         1         2         3
12345678901234567890123456789012
270                     32000 C
310DECARLO RICHARD      30000HA
610                           D
688SMITH    GERRY       A00525HA
791                      00475 C
828GOYAK    DEBBIE      A34000SA
857                           D
```

Diseño de un informe de auditoría significativa con la cuenta adecuada. Antes de actualizar el archivo, compruebe los siguientes límites:

```
Rate type       Minimum Rate      Maximum Rate
H               $4.00             $10.00
S              $200.00           $500.00
```

7. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos). Los siguientes tabla describe las transacciones utilizadas para actualizar el costo y el precio de venta (sólo) para la herramienta de archivo:

```
Field        Field
Nbr    Name  Description    Begins  Ends  Len  Format
1 TID   Tool ID              1     3    3 CH
2 TCOST Tool cost each       4     8    5 999V99
3 TSELL Tool sells for       9    13    5 999V99
4 CRLF  PC/370 Only         14    15    2 CR/LF
```

Escriba un programa que se aplicará a las operaciones siguientes a la Herramienta de archivo. Tenga en cuenta que no hay códigos de operación: los registros se puede cambiar (únicamente). Por supuesto, un registro debe existir antes de que se puede cambiar ...

```
         1
1234567890123
H8       01099
PLM00375
PLX      00599
SPM0025000419
SSM0041900250
SSS00399
WSP00025
```

Diseño de un informe de auditoría significativa con la cuenta adecuada. Antes de actualizar el archivo, verificar que el costo es menor que el precio de venta, con excepción de las envolturas, que debe tener una venta de precio de cero. (Una envoltura está indicado por una W en la primera posición de la herramienta de identificación.)

Capítulo 13

Más Bolsas de aritmética decimal

Objetivos Al término de este capítulo usted será capaz de:

- Utilice la instrucción MP para multiplicar un número por otro,
- Utilice la instrucción de DP para dividir un número por otro,
- Mostrar la forma de evitar la división por cero,
- Mostrar la forma de seguimiento de la ubicación del punto decimal implícita cuando se utiliza MP y DP,
- Utilice la instrucción de SRP multiplicar o dividir un número por una potencia de diez, y
- Utilice la instrucción de SRP para redondear un número, especialmente después de un cociente de la división.

Introducción En el capítulo siete se introdujo el formato decimal empaquetado, así como las instrucciones seleccionados incluyendo AP, SP, CP, ZAP, empacar y UNPK. En este capítulo veremos la multiplicación (MP), dividir (DP), y el cambio-y-vuelta (SRP) de instrucciones.

En este capítulo vamos a continuar nuestro trabajo con el archivo de inventario para las Industrias Dindón. Recall el diseño de registro para ese archivo, COGS.DAT, es el siguiente:

```
          Campo Campo
N º Nombre Descripción Inicio Término Len Formato
1 DESC Producto desc 1 10 10 CH
2 CALIF ventas California 11 13 3 ZD
3 ILL ventas Illinois 14 16 3 ZD
4 UTAH ventas de Utah 17 19 3 ZD
```

```
5 WISC ventas de Wisconsin 20 22 3 ZD
6 COMIENZAN A partir inv. 23 25 3 ZD
7 Compras purch 26 28 3 ZD
8 QOH Cantidad a mano 29 31 3 ZD
9 Coste COSTE (cada uno) 32 35 4 99V99
10 Vendo Vendo por (cada uno) 36 39 4 99V99
11 CRLF PC/370 Sólo 40 41 2 CR / LF
```

... y los datos es la siguiente:

```
.        1         2         3
12345678901234567890123456789012345678 9
GIZMOS     020030020020017099023122529999
WIDGETS    015010010002022034019001100 25
JUNQUE     025015015018030052010015503 39
```

Veremos dos programas en este capítulo. El primero dice COGS.DAT y calcula y muestra la empresa ventas en dólares de ancho. La salida se realiza a través de WTOs sólo: no hay ningún archivo de salida. El resultado es el siguiente:

```
A:\MIN> cogs13a
        COGS13A ... Begin execution
        COGS13A ... Nationwide dollar sales are 2,955.82
        COGS13A ... Normal end of program
```

Veamos el cálculo de las ventas en dólares en todo el país:

```
Product Calif Ill Utah Wisc Total Sell for Product
GIZMOS    20  30    20   20   90   29.99   2699.10
WIDGETS   15  10    10    2   37     .25     9.25
JUNQUE    25  15    15   18   73    3.39    247.47
                                 TOTAL   2955.82
```

Como puede ver, tendremos que multiplicar el fin de determinar la venta de dólares a nivel nacional. Nosotros Ahora analizaremos la Bolsas de multiplicar la enseñanza en general, después de que volveremos a este problema.

13.1. Las instrucciones que Multiplicar

El formato general de las Bolsas de Multiply (MP) la instrucción es: MP FLDA, FLDB

Esta instrucción se multiplicarán FLDA por FLDB almacenar el producto (resultado) en FLDA. Por supuesto, tanto los campos deben ser válidos campos de pic nic. Hay también dos otras normas muy importantes:

- La longitud del segundo operando debe ser menor o igual a 8, y

- El primer operando debe tener tantos bytes de alto ceros orden, ya que hay bytes (todos los bytes) en el segundo operando. (Tenga en cuenta que sigue, por tanto, que el segundo operando debe ser menor en longitud que el primer operando.)

Esta segunda regla es algo confuso y se explica mejor a través de algunos ejemplos. Habida cuenta de la siguientes definiciones:

```
FLDA DC PL3'5'
FLDB DC PL3'20'
FLDC DC PL3'1000'
FLDD DC PL5'5'
FLDE DC PL5'20'
FLDF DC PL5'1000'
```

Tenga en cuenta la multiplicación de las instrucciones siguientes:

1. MP FLDA, FLDB

   Recordemos que la longitud del primer operando (FLDA aquí) debe ser mayor que la longitud de la el segundo operando (FLDB aquí). Dado que estos campos son de igual longitud (ambos son PL3), este MP no es válido. Tenga en cuenta que esto es a pesar del hecho de que el producto $(5 * 20 = 100)$ se ajuste en un Campo PL3!

   Nota: Esta instrucción se reunirán correctamente. El error no se detecta hasta que ejecute tiempo! Esto puede ser visto con el siguiente (muy poco) del programa:

```
MP   BEGIN
     MP     FLDA,FLDB
     RETURN
FLDA DC     PL3'5'
FLDB DC     PL3'20'
FLDC DC     PL3'1000'
FLDD DC     PL5'5'
FLDE DC     PL5'20'
FLDF DC     PL5'1000'
     END
```

   (Debe clave de este programa y ejecutarlo. Al cambiar la línea MP sólo, puede probar cada uno de de las instrucciones MP estaremos mirando y verificar las conclusiones presentadas en en todo el capítulo.)

   De las imágenes de pantalla siguiente se puede ver que, como se mencionó anteriormente, el error no es detectado hasta el tiempo de ejecución:

```
A:\MIN>l370 mp/lx
        (Copyright message appears here)
```

```
PC/370 LINKAGE EDITOR OPTIONS ON = LXEFIP
STATS SYM=00001 MAXSTD=00001 BTMEM=57788
NO ERRORS FOUND
A:\MIN>mp
TRACE EP A=08E4 ID=BUG
         370 A=000270 OP=58DD0004
  ********************************************
  *    PC/370 System Release 4.2          01/07/88    *
  *    Copyright (C) 1988 Donald S. Higgins           *
  *                                                   *
  * You are encouraged to copy and share this         *
  * package with other users on the condition         *
  * the package is not distributed in modified        *
  * form, and that no fee is charged. If you          *
  * find PC/370 useful, send 45 dollars to the        *
  * address below to become registered user and       *
  * support continued shareware development.          *
  * Registered users will receive notices of          *
  * future PC/370 releases.                           *
  *                                                   *
  *       Don Higgins                                 *
  *       6365 - 32 Avenue, North                     *
  *       St. Petersburg, Florida 33710               *
  ********************************************
TYPE H FOR HELP +
```

Al igual que en nuestro análisis anterior de PC/370 's laboratorio, ID = ERROR nos dice que el programa ha composición, y A = 000270 nos dice dónde. Restamos X'200 'de la dirección que nos da 000070, la dirección de la siguiente instrucción a ejecutar el programa no había terminado. Nosotros ver en la lista. PRN que esta es la ubicación de la macro de regreso, para la instrucción lo que provocó el error es la instrucción anterior:

```
MP                                                    PAGE    1
PC/370  CROSS ASSEMBLER   OPTIONS=LXACE

LOC                ADR1   ADR2 LINE  LABEL    OP        OPERANDS
000000                                   1 *+       BEGIN
000000                                   2 MP       CSECT
000000                                   3          USING     *,15
000000  47F0F058               0058      4          B         KZHQX001
000004  0B                               5          DC        AL1(11)
000005  D4D7404040404040                 6          DC        CL11'MP       '
000010  0000000000000000                 7 HZQKX001 DC        18F'0'
000058  90ECD00C               000C      8 KZHQX001 STM       14,12,12(13)
```

```
00005C  50D0F014                0014     9          ST        13,HZQKX001+4
000060  18ED                             10          LR        14,13
000062  41D0F010                0010    11          LA        13,HZQKX001
000066  50D0E008                0008    12          ST        13,8(0,14)
00006A                                  13          DROP      15
00006A                                  14          USING     HZQKX001,13
00006A  FC22D06AD06D    007A    007D    15          MP        FLDA,FLDB
000070                                  16  ******** RETURN
000070  58DD0004                0004    17          L         13,4(13)
000074  98ECD00C                000C    18          LM        14,12,12(13)
000078  07FE                            19          BR        14
00007A  00005C                          20  FLDA    DC        PL3'5'
00007D  00020C                          21  FLDB    DC        PL3'20'
000080  01000C                          22  FLDC    DC        PL3'1000'
000083  000000005C                      23  FLDD    DC        PL5'5'
000088  000000020C                      24  FLDE    DC        PL5'20'
00008D  000001000C                      25  FLDF    DC        PL5'1000'
000098                                  26          END
```

2. MP FLDD,FLDC

   La longitud del segundo operando es menor que el FLDD longitud del primer operando. Esta es una condición necesaria pero no suficiente para el MP. Let's vistazo a la representación hexadecimal del primer operando. FLDD es X'000000005C. Desde FLDC es PL3, FLDD debe tener al menos tres bytes de alto orden ceros, es decir, seis dígitos hexadecimales de cero como en X'000000. Podemos ver que FLDD de hecho tiene cuatro bytes de ceros de orden superior. Por lo tanto, este MP es válido.

   Una vez más, podemos comprobar esto mediante el uso PC/370 's laboratorio para detener la ejecución de MP.MLC inmediatamente después de la MP, y luego ver FLDD. Debemos ver $5 * 1000 = 5000$. Recuerde que una T mayúscula es necesario para utilizar el laboratorio. El examen completo se muestra aquí con respuestas de los usuarios en negrita:

```
A:\MIN>mp
T TRACE EP A=07AB ID=370              370 A=000200 OP=47F0F058
         (Copyright message appears here)
TYPE H FOR HELP
+a
ADDR STOP ON
A=270
      000270 58DD0004 98ECD00C 07FE0000 5C00020C ....q.......*...
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET
TRACE EP A=1433 ID=BC               370 A=000200 OP=47F0F058
TRACE EP A=1F9B ID=STM              370 A=000258 OP=90ECD00C
```

```
TRACE EP A=17D1 ID=ST                370 A=00025C OP=50D0F014
TRACE EP A=0CAD ID=LR                370 A=000260 OP=18ED
TRACE EP A=1649 ID=LA                370 A=000262 OP=41D0F010
TRACE EP A=17D1 ID=ST                370 A=000266 OP=50D0E008
TRACE EP A=2376 ID=MP                370 A=00026A OP=FC42D073D070
ADDR STOP
      000270 58DD0004 98ECD00C 07FE0000 5C00020C ....q.......*...
TRACE EP A=162D ID=L                 370 A=000270 OP=58DD0004
+d
A=280
      000280 01000C00 0005000C 00000002 0C000001 ................
      000290 000C0000 00000000 00000000 0007C6E8 ..............FY
+
(Press Esc to return to the DOS prompt.)
```

3. MP FLDF, FLDA

   es PL5'1000 y FLDA es PL3'5. Esto es similar al del ejemplo anterior donde se FLDF multiplicar 5 por 1000, pero aquí se multiplican 1000 por 5. Intuitivamente, si el ejemplo anterior es válida, entonces también lo debe ser éste. Pero este no es el caso! Una vez más, echemos un vistazo a la hexagonal la representación del primer operando. FLDF es X'000001000C. Desde FLDA es PL3, FLDF debe tener al menos tres bytes de alto orden ceros, es decir, seis dígitos hexadecimales de cero, como en X'000000. Podemos ver que FLDF tiene sólo cinco ceros de alto orden. Por lo tanto, esta MP es válido.

   Verificar esta utilizando el dispositivo de prueba.

intenta

1. MP demostrar por qué no es válida.

   ```
   FLDB, FLDE 2.
   ```

2. Mostrar por qué

   ```
   MP FLDE, FLDB
   ```

   es válido.

3. Mostrar por qué

   ```
   MP FLDE, FLDC
   ```

   es válido.

4. Mostrar por qué

   ```
   FLDB MP = PL1'5 '
   ```

es válido.

5. Mostrar por qué

```
MP FLDB, = PL2'10 '
```

no es válida.

6. Mostrar por qué

```
FLDE MP = PL2'10 '
```

es válido.

7. Mostrar por qué

```
MP FLDE, = CL2'10 '
```

no es válida.

13.2. Programa de ejemplo: las ventas en dólares a nivel nacional Dindón's

Volvamos ahora a nuestro problema de programación original: queremos determinar de dólares en todo el país ventas. Nuestra producción aparecerá como sigue:

```
A:\MIN>cogs13a
COGS13A ... Begin execution
COGS13A ... Nationwide dollar sales are 2,955.82
COGS13A ... Normal end of program
```

... que nos aparecerá con una de la OMC, por lo que define el resultado como:

```
OMSG     DS 0CL49
         DC CL39'COGS13A ... Nationwide dollar sales are'
ODOLLARS DC XL10'4020206B2021204B2020'   BZZ,ZZ9.99
```

De dólares de salida será de hasta 99,999.99, para un total de 7 dígitos, por lo que vamos a utilizar un campo de PL4 de se acumulan las ventas. Por lo tanto, definir este total como:

```
WDOLLARS DC PL4'0' Nationwide dollar sales
```

Vamos a determinar las ventas totales dentro de la sección de proceso. Para cada registro (que contiene unidades vendidos por producto), las ventas de cada estado, se añade, dando el total de unidades vendidas para este producto. El total de de unidades vendidas se multiplica por el precio de venta del producto, dando las ventas totales en dólares para este producto. Esto cifra se añade a la del acumulador. Leer el registro siguiente y repetir.

```
PROCESS EQU  *
        ST   R10,SVPROC
        PACK WCALIF,ICALIF Each product's sales must
        PACK WILL,IILL      be packed so they can be
        PACK WUTAH,IUTAH     added to total for this
        PACK WWISC,IWISC      product...
        ZAP  WTOTAL,=P'0'  Initialize the total to zero
        AP   WTOTAL,WCALIF  and start adding...
        AP   WTOTAL,WILL
        AP   WTOTAL,WUTAH
        AP   WTOTAL,WWISC
        PACK WSELL,ISELL   Unit sell price
        ZAP  PK5,WTOTAL    Length of WTOTAL is PL2
        MP   PK5,WSELL      and length of WSELL is PL3
        AP   WDOLLARS,PK5    so need PL5 for product.
        BAL  R10,READ
        L    R10,SVPROC
        BR   R10
```

El programa completo, COGS13A.MLC, sigue.

314 ⟨*search..., read INVENTRY into IREC, EOFSW, SVREAD return* 314⟩≡ (315 353 357 418 421)

```
READ    EQU   *
        ST    R10,SVREAD
        GET   INVENTRY,IREC              Read a single product record
        B     READX
ATEND   EQU   *
        MVI   EOFSW,C'Y'
READX   EQU   *
        L     R10,SVREAD
        BR    R10
```

315 ⟨*COGS13A.MLC* 315⟩≡

```
                PRINT NOGEN
        ****************************************************************
        *       FILENAME: COGS13A.MLC                                  *
        *       AUTHOR : Bill Qualls                                   *
        *       SYSTEM : PC/370 R4.2                                   *
        *       REMARKS : Determine nationwide dollar sales for        *
        *                     COGSWORTH INDUSTRIES.                    *
        ****************************************************************
                START 0
                REGS
        BEGIN   BEGIN
                WTO   'COGS13A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
                WTO   'COGS13A ... Normal end of program'
                RETURN
        ****************************************************************
        *       SETUP - Those things which happen one time only,       *
        *                 before any records are processed.            *
        ****************************************************************
        SETUP   EQU   *
                ST    R10,SVSETUP
                OI    INVENTRY+10,X'08' PC/370 ONLY - Convert all
        *                                       input from ASCII to EBCDIC
                OPEN INVENTRY
                BAL   R10,READ
                L     R10,SVSETUP
                BR    R10
****************************************************************
*       READ - Read a record.                                  *
****************************************************************
⟨search..., read INVENTRY into IREC, EOFSW, SVREAD return 314⟩
        ****************************************************************
        *       PROCESS - Those things which happen once per record.   *
        ****************************************************************
        PROCESS EQU   *
                ST    R10,SVPROC
                PACK WCALIF,ICALIF              Each product's sales must
                PACK WILL,IILL                    be packed so they can be
                PACK WUTAH,IUTAH                    added to total for this
                PACK WWISC,IWISC                     product...
                ZAP   WTOTAL,=P'0'              Initialize the total to zero
                AP    WTOTAL,WCALIF               and start adding...
                AP    WTOTAL,WILL
                AP    WTOTAL,WUTAH
                AP    WTOTAL,WWISC
```

```
         PACK WSELL,ISELL                Unit sell price
         ZAP   PK5,WTOTAL                Length of WTOTAL is PL2
         MP    PK5,WSELL                   and length of WSELL is PL3
         AP    WDOLLARS,PK5                  so need PL5 for product.
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         ED    ODOLLARS,WDOLLARS
         WTO   OMSG
         CLOSE INVENTRY
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP DC     F'0'                    SETUP
SVPROC   DC    F'0'                    PROCESS
SVREAD   DC    F'0'                    READ
SVWRAP   DC    F'0'                    WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFSW    DC    CL1'N'                  End of file? (Y/N)
WCALIF   DC    PL2'0'                  Units sold in Calif
WILL     DC    PL2'0'                  Units sold in Illinois
WUTAH    DC    PL2'0'                  Units sold in Utah
WWISC    DC    PL2'0'                  Units sold in Wisconsin
WTOTAL   DC    PL2'0'                  Units sold in all states
WSELL    DC    PL3'0'                  Sell for (each) 999V99
WDOLLARS DC    PL4'0'                  Nationwide dollar sales
PK5      DC    PL5'0'
```

```
*****************************************************************
*        Input record definition                                *
*****************************************************************
IREC     DS    0CL41            1-41 Inventory record
IDESC    DS    CL10             1-10 Product description
ICALIF   DS    CL3             11-13 Units sold in Calif
IILL     DS    CL3             14-16 Units sold in Illinois
IUTAH    DS    CL3             17-19 Units sold in Utah
IWISC    DS    CL3             20-22 Units sold in Wisconsin
IBEGIN   DS    CL3             23-25 Beginning inventory
IPURCH   DS    CL3             26-28 Purchases throughout year
IQOH     DS    CL3             29-31 Actual quantity on hand
ICOST    DS    CL4             32-35 Cost (each) 99V99
ISELL    DS    CL4             36-39 Sell for (each) 99V99
ICRLF    DS    CL2             40-41 PC/370 only - CR/LF
*****************************************************************
*        Output message definition                              *
*****************************************************************
OMSG     DS    0CL49
         DC    CL39'COGS13A ... Nationwide dollar sales are'
ODOLLARS DC    XL10'4020206B2021204B2020'   BZZ,ZZ9.99
         END   BEGIN
```

13.3. Redondeo con SRP

Considere el siguiente problema de programación: Un archivo de imagen de la tarjeta contiene un precio en las posiciones 16-20 (999V99) y un por ciento de descuento en el valor de las posiciones 21-22 (99V). Queremos determinar la venta Precio (BZZ9.99). Vamos a utilizar las definiciones de campo siguientes:

```
PRICE  DS CL5
PCTOFF DS CL2
SALE   DC X'402021204B2020'
```

Suponga PRECIO contiene '00425 ', lo que representa 4,25 dólares, y PCTOFF contiene '15', lo que representa el 15 %. En primer lugar, determinar el descuento como x PCTOFF PRECIO: $ 4.25 x .15 = $ .6375. Sin embargo, este descuento, .6375, Debe ser redondeado a .64. Luego restamos el (redondeado) de descuento del precio regular dando el precio de venta: $ 4.25 - $ .64 = $ 3.61. Ahora veremos el redondeo, después de que volveremos a este problema.

* * * * * * * * * * * * * * * * * * *

El redondeo en BAL se realiza a través de la SRP (Shift-y todo el relleno) la instrucción. Esta instrucción puede ser usado para algo más que el redondeo. ¿Cómo funciona es bastante interesante. Tenga en cuenta la siguientes:

```
FLDA DC PL4'1278' lo mismo que: X'0001278C'
```

La instrucción de las causas SRP los datos en el primer operando a ser desplazado a la izquierda oa la derecha por el número de dígitos especificado en el segundo operando. Por ejemplo,

```
SRP FLDA, 1,0 daría: X'0012780C'
                  ::
                  : :..... Este cero dice que no todo el año.
                  : (La opción de redondeo no
                  : Aplicación a la multiplicación.)
                  :
                  :........ Éste dice que cambia a otro
                            dígitos a la izquierda.
```

Tenga en cuenta que el resultado, X'0012780C 'es igual a PL4'12780', o 12.780, que es el mismo que si nos FLDA se había multiplicado por diez! Otro ejemplo (utilizando los datos originales):

```
 SRP FLDA, 3,0 daría: X'1278000C'
                  :
                  :........ Este tres dice a cambio de tres
                            dígitos a la izquierda.
```

Tenga en cuenta que el resultado, X'1278000C 'es igual a PL4'1278000', o 1.278.000, que es el mismo que si FLDA que se habían multiplicado por mil! Por supuesto, 1000 = 10 3 por lo tanto el 3 como el segundo operando, y, en general, SRP con un segundo operando de n es equivalente a multiplicar por el campo 10n. Otro ejemplo:

```
SRP FLDA, 62,0 daría: X'0000012C'
                  ::
                  : :.... Este cero dice que no todo el año.
                  :
                  :........Este 62, que es 64-2, dice que
                           dígitos cambio de dos a la derecha.
```

Tenga en cuenta que el resultado, X'0000012C 'es igual a PL4'12', o 12, que es lo mismo que si tuviéramos que dividido FLDA por cien! Continuando con los datos originales:

```
SRP FLDA ,64-2, 5 daría: X'0000013C'
                  ::
                  :: .. Este cinco dice hacer todo el año.
                  :
                 :........ Esto es a menudo usado en lugar
                             de 62 en el ejemplo anterior.
```

Tenga en cuenta que el resultado, X'0000013C 'es igual a PL4'13', o 13, que es lo mismo que si tuviéramos que FLDA dividido por diez (da 127), añadió cinco (da 132), y dividido por diez (dando 13).

En general, SRP con un segundo operando de la forma 64-N es equivalente a multiplicar por el campo 10-N, o dividir el campo por 10n. Si el tercer operando es cero, el resultado no se redondea. Si el tercer operando es de cinco, se redondea el resultado.

* * * * * * * * * * * * * * * * * * * *

Volvemos a nuestro problema de programación. Teniendo en cuenta:

```
PRICE  DS CL5        contains 00425 representing $4.25
PCTOFF DS CL2        contains 15 representing 15%
SALE   DC X'402021204B2020'
```

Tanto los primeros campos deberán ser envasados. Podemos Pack PCTOFF en el campo de dos bytes. PRECIO caben en un tres campos de bytes, pero si vamos a multiplicarlo por el por ciento de descuento (a dos campos de bytes), entonces debe definirse como el campo de cinco bytes:

```
PACK PK5,PRICE
PACK PK2,PCTOFF
MP   PK5,PK2    Multiply PRICE by PCTOFF
SRP  PK5,64-2,5  giving DISCOUNT (rounded)
PACK PK3,PRICE  PRICE minus
SP   PK3,PK5     DISCOUNT equals
ED   SALE,PK3     SALE PRICE
```

intenta Teniendo en cuenta:

```
X DC PL3'456'
Y DC PL4'2345'
```

1. 8. Demostrar que SRP X, 2,0 da X'45600C '
2. 9. Demostrar que SRP Y, 3,0 da X'2345000C '
3. 10. Mostrar X después de SRP X, 63,0
4. 11. Mostrar Y después de SRP Y ,64-2, 0
5. 12. Mostrar X después de SRP X, 62,0
6. 13. Mostrar X después de SRP X, 62,5
7. 14. Mostrar Y después de SRP Y ,64-3, 0
8. 15. Mostrar Y después de SRP Y ,64-3, 5

13.4. Muestra del Programa: Contribución de California a las ventas de

Haremos uso de la instrucción de SRP en nuestro próximo programa, que es producir un informe que muestra el porcentaje de las ventas por producto para California, es decir, la contribución de California a un total de ventas. El informe aparecerá como sigue:

```
1         2         3         4           5       6
12345678901234567890123456789012345678901234567890123456789012345678901234567890
              COGSWORTH INDUSTRIES
       California's Contribution to Sales
              Nationwide   California    Percent of
 Product         Sales         Sales      National
----------    ----------   ----------    ----------
XXXXXXXXXX       BZZ9          BZZ9        BZZ9%
XXXXXXXXXX       BZZ9          BZZ9        BZZ9%
XXXXXXXXXX       BZZ9          BZZ9        BZZ9%
----------    ----------   ----------    ----------
TOTALS           BZZ9          BZZ9        BZZ9%
```

Con el fin de determinar el porcentaje de las ventas por producto para California, dividimos las ventas de California total (nacional) de ventas. Ahora analizaremos la brecha Bolsas de instrucción en general, después de que Volveremos a este problema.

13.4.1. La Instrucción DP

En el desempeño de cualquier aritmética en BAL, debemos tener en cuenta que no existe una asignación por punto decimal, es decir, toda la aritmética BAL se presume que es la aritmética de enteros. Depende de usted, como el programador, no perder de vista que el punto decimal se van, y puso allí al imprimir el número. Esto es aún más complicada en el caso de la división. Toda división BAL es división de módulo, es decir, el resultado de una división es siempre un cociente entero y un entero , el resto nunca un número fraccionario.

Una revisión de la terminología de la división es adecuada. Si se dividen 25 por 4, tengo 6 y un resto de 1. El dividendo es 25, el divisor es 4, y el cociente es de 6.

```
Dividend --> 25      6  <-- Quotient
Divisor -->   4 4 | 25
                  24
                   1 <-- Remainder
```

Eso es lo que es como en BAL: Como resultado de una división, se obtiene un cociente y un resto. El formato de la instrucción de la brecha de relleno es:

```
DP dividend,divisor
```

Ambos campos deben ser válidos números empaquetados, o bien el programa Abend. A raíz de la brecha (esta parte es muy raro ...) el área que contiene el dividendo se divide en dos partes: el resto se coloca en el lado derecho de dividendos. Su longitud es igual a la del divisor. El cociente es luego se coloca en el lado izquierdo de los dividendos en los bytes no ocupada por el resto. Para ejemplo:

```
             dividend
quotient                remainder
what's left           length of divisor
```

Por ejemplo, dadas las definiciones de los campos siguientes:

```
A DC PL5'25' dividend
B DC PL2'4'  divisor
```

... y luego resulta en lo siguiente:

```
A, before DP 00 00 00 02 5C A,
after DP  00 00 6C 00 1C
```

El cociente es, entonces, en A (3) y el resto está en A 3 (2). Tenga en cuenta que A, en su totalidad, no es de ser un número válido de pic nic.

intenta

1. 16. Dado A PL3'47 DC y B DC PL1'9 ', muestran que DP A, los resultados de B en A = '005C2C'. ZAP la cociente en Q y ZAP el resto en R.
2. 17. Dado C DC PL5'1276 y D DC PL2'100 ', muestran C y D después de DP C, D. ZAP el cociente en Q ZAP y el resto en R.
3. 18. Habida cuenta de E PL4'10 DC y DC PL2'25 F, E y F, después de mostrar DP E, F. ZAP el cociente en Q y ZAP el resto en R.

Los resultados del ejercicio anterior se puede demostrar a través de un programa corto y PC/370 's laboratorio. Considere el siguiente programa, DP.MLC:

```
DP BEGIN
   DP     A,B
   DP     C,D
   DP     E,F
   RETURN
A  DC     PL3'47'
B  DC     PL1'9'
C  DC     PL5'1276'
D  DC     PL2'100'
E  DC     PL4'10'
F  DC     PL2'25'
   END
```

El resultado de archivo. PRN es el siguiente:

```
DP                                                      PAGE    1
PC/370  CROSS ASSEMBLER   OPTIONS=LXACE
   LOC                  ADR1   ADR2 LINE  LABEL    OP        OPERANDS
000000                                    1 *+       BEGIN
000000                                    2 DP       CSECT
000000                                    3          USING     *,15
000000  47F0F058                 0058     4          B         KZHQX001
000004  0B                                5          DC        AL1(11)
000005  C4D7404040404040                  6          DC        CL11'DP       '
000010  0000000000000000                  7 HZQKX001 DC        18F'0'
000058  90ECD00C                 000C     8 KZHQX001 STM       14,12,12(13)
00005C  50D0F014                 0014     9          ST        13,HZQKX001+4
000060  18ED                            10          LR        14,13
000062  41D0F010                 0010    11          LA        13,HZQKX001
000066  50D0E008                 0008    12          ST        13,8(0,14)
00006A                                  13          DROP      15
00006A                                  14          USING     HZQKX001,13
00006A  FD20D076D079    0086    0089    15          DP        A,B
000070  FD41D07AD07F    008A    008F    16          DP        C,D
000076  FD31D081D085    0091    0095    17          DP        E,F
00007C                                  18 *+++++++ RETURN
00007C  58DD0004                 0004    19          L         13,4(13)
000080  98ECD00C                 000C    20          LM        14,12,12(13)
000084  07FE                            21          BR        14
000086  00047C                          22 A        DC        PL3'47'
000089  9C                              23 B        DC        PL1'9'
00008A  000001276C                      24 C        DC        PL5'1276'
00008F  100C                            25 D        DC        PL2'100'
000091  0000010C                        26 E        DC        PL4'10'
000095  025C                            27 F        DC        PL2'25'
000098                                  28          END
```

Vamos a utilizar la facilidad de texto para detener el programa antes y después de la AD y ver el contenido de la A, B, C, D, E y F. Los resultados son los siguientes:

```
A:\MIN>dp
T TRACE EP A=07AB ID=370         370 A=000200 OP=47F0F058
       (Copyright message appears here)
TYPE H FOR HELP
+a
ADDR STOP ON
A=26a
    00026A FD20D076 D079FD41 D07AD07F FD31D081 .....‘...:."...a
```

```
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET TRACE EP A=1433 ID=BC          370 A=000200 OP=47F0F058
TRACE EP A=1F9B ID=STM          370 A=000258 OP=90ECD00C
TRACE EP A=17D1 ID=ST           370 A=00025C OP=50D0F014
TRACE EP A=0CAD ID=LR           370 A=000260 OP=18ED
TRACE EP A=1649 ID=LA           370 A=000262 OP=41D0F010
TRACE EP A=17D1 ID=ST           370 A=000266 OP=50D0E008
ADDR STOP
    00026A FD20D076 D079FD41 D07AD07F FD31D081 .....‘...:."...a
TRACE EP A=2110 ID=DP           370 A=00026A OP=FD20D076D079
+d
               | A |B|            C       | D | E    | F
A=286     000286 00047C9C 00000127 6C100C00 00010C02 ..@.....%.......
    000296 5C 000000 00000007 C6E80000 00000000 *.......FY......
+a
ADDR STOP OFF
+a
ADDR STOP ON
A=27c
    00027C 58DD0004 98ECD00C 07FE0004 7C9C0000 ....q.......@...
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET
TRACE EP A=2110 ID=DP           370 A=000270 OP=FD41D07AD07F
TRACE EP A=2110 ID=DP           370 A=000276 OP=FD31D081D085
ADDR STOP
    00027C 58DD0004 98ECD00C 07FE005C 2C9C0001 ....q......*....
TRACE EP A=162D ID=L            370 A=00027C OP=58DD0004
+d
              | A |B|            C       | D | E    | F A=286
    000286 005C2C9C 00012C07 6C100C00 0C010C02 .*......%.......
    000296 5C 000000 00000007 C6E80000 00000000 *.......FY......
+
(Press Esc to return to the DOS prompt.)
```

* * * * * * * * * * * * * * * * * * * *

La instrucción DP tiene normas similares a las de la MP:

- La longitud del segundo operando debe ser menor o igual a 8, y
- La longitud del segundo operando debe ser menor que la longitud del primer operando.

Además, de acuerdo con los principios de IBM de operaciones:

> "A excepción de división decimal se produce si el dividendo no tiene al menos un cero a la izquierda ... A el cociente mayor que

el número de dígitos permitidos se reconoce como una excepción de división decimal. La operación se suprime, y se produce una interrupción del programa. "

Por ejemplo, el programa siguiente Abend:

```
DP2       BEGIN
          DP     X,Y
          RETURN
X         DC     PL5'5006'
Y         DC     PL3'5'
          END
Dividend -->   5006            1001 <-- Quotient
Divisor -->     5          5 | 5006
                              5005
                                  1 <-- Remainder
  X, before DP            00   00    05   00   6C
  X, after DP             00   1C    00   00   1C
                         |__ not enough room for quotient ( 1001)
```

Es cierto que estas normas de mantenimiento de recta, y recordando que es el cociente y dónde está el resto puede ser confuso. Así que recomiendo la siguiente "prueba de tonto" método para hacer ....

división Yo siempre definir los siguientes campos para la división:

```
DIVIDEND DS 0PL16
QUOTIENT DS PL8
REMAINDR DS PL8
DIVISOR  DS PL8
```

Antes de la división, voy a mover el dividendo a dividendo y el divisor de divisor. Luego de verificación para Asegúrese de que el divisor no es cero, como la división por cero es prohibido. Luego realizar la DP, que deja el cociente en cociente y el resto en REMAINDR. Puedo cambiar y la vuelta de la COCIENTE en su caso, a continuación, ZAP en el campo de destino. Considere el siguiente ejemplo. Teniendo en cuenta

```
COUNT   DS PL3
TOTAL   DS PL3
PERCENT DS CL4 BZZ9
```

Suponga COUNT = 200 y TOTAL = 300. Luego, cuente divididos por el total es .66666, o 67 % después de redondeo. Me código de la siguiente manera:

```
ZAP DIVIDEND,COUNT
ZAP DIVISOR,TOTAL
CP  DIVISOR,=P'0'
BE  error
```

```
SRP DIVIDEND,3,0         Dividend becomes 200000
DP  DIVIDEND,DIVISOR     Quotient becomes 666
SRP QUOTIENT,64-1,5      Quotient becomes 67
MVC PERCENT,=X'40202120' Three digits only so use
ED  PERCENT,QUOTIENT+6   last two bytes of quotient
```

13.4.2. Comentarios

1. 1. El código de condición se establece como resultado de ZAP. El código de estado indica si el resultado es cero, menos de cero, mayor que cero, o desbordamiento. Por consiguiente, no es necesario hacer el PP después de la ZAP. En su lugar, podría el siguiente código:

   ```
   ZAP DIVISOR,TOTAL
   BZ  error
   ```

2. No me gusta el desplazamiento explícito, por lo que prefiere utilizar un campo de trabajo para la ED:

   ```
   MVC PERCENT,=X'40202120' Three digits only so use
   ZAP PK2,QUOTIENT         last two bytes of quotient
   ED  PERCENT,PK2
   ```

Por supuesto, sería bueno tener una manera fácil de incluir estos campos de la división en un programa de sin introducir ellos o copiar de otro programa. Bueno, ahí está! Podemos hacer uso de PC/370 's función de copia. Simplemente, crear un archivo que contiene las instrucciones que desea copiar. Esto archivo debe tener la extensión. CPY. Aquí está mi DIVISION.CPY:

```
*************************************************************
* This is DIVISION.CPY - BQ's Easy Divides                  *
* Usage: COPY DIVISION (with COPY in column 10)             *
*************************************************************
DIVIDEND DS 0PL16
QUOTIENT DS PL8
REMAINDR DS PL8
DIVISOR  DS  PL8
```

Luego, en mi programa, la División I COPIA tipo (con copia en la columna 10). En el tiempo de montaje, el archivo de con ese nombre (y extensión. CPY) se fusionó con el. MLC archivo que es la entrada a la M370 programa, creando el archivo. ALC archivo que entra en el programa A370.

Nota: los miembros de copia no pueden contener macros o miembros de la otra copia: el programa no M370 de verificación de "anidado" declaraciones COPIA, o macros en las declaraciones de copia.

son útiles no sólo para las definiciones de campo común de este tipo, pero para los diseños de registro de COPIA como así! Por ejemplo, puedo crear COGS.CPY como sigue:

```
      ***************************************************************
              *         This is COGS.CPY - Cogsworth's Inventory Data
              *         Usage: COPY COGS                (with COPY in column 10)
              ***************************************************************
              IREC      DS      0CL41             1-41 Inventory record
              IDESC     DS      CL10              1-10 Product description
              ICALIF    DS      CL3              11-13 Units sold in Calif
              IILL      DS      CL3              14-16 Units sold in Illinois
              IUTAH     DS      CL3              17-19 Units sold in Utah
              IWISC     DS      CL3              20-22 Units sold in Wisconsin
              IBEGIN    DS      CL3              23-25 Beginning inventory
              IPURCH    DS      CL3              26-28 Purchases throughout year
              IQOH      DS      CL3              29-31 Actual quantity on hand
              ICOST     DS      CL4              32-35 Cost (each) 99V99
              ISELL     DS      CL4              36-39 Sell for (each) 99V99
              ICRLF     DS      CL2              40-41 PC/370 only - CR/LF
```

* * * * * * * * * * * * * * * * * * * *

Volvamos ahora a nuestro problema de programación de la segunda para determinar el porcentaje de las ventas por producto para California:

```
      1         2         3         4         5         6
              12345678901234567890123456789012345678901234567890123456789012345678901234567890
                                      COGSWORTH INDUSTRIES
                              California's Contribution to Sales
                                      Nationwide         California           Percent of
                        Product           Sales              Sales             National
                        ----------      ----------         ----------          ----------
                        XXXXXXXXXX          BZZ9               BZZ9               BZZ9%
                        XXXXXXXXXX          BZZ9               BZZ9               BZZ9%
                        XXXXXXXXXX          BZZ9               BZZ9               BZZ9%
                        ----------      ----------         ----------          ----------
                        TOTALS              BZZ9               BZZ9               BZZ9%
```

Tenga en cuenta que la línea de partida segundo contiene un apóstrofo. Para incluir un apóstrofo en un literal, el código de un apóstrofo doble (no una cita). Por ejemplo:

```
              HD2       DS      0CL62
                        DC      CL60'           California''s Contribution to Sales'
                        DC      XL2'0D25'
```

Con el fin de determinar el porcentaje de las ventas por producto para California, se divide por el total de ventas de California (nacional) de ventas. Después de encontrar las ventas totales de este producto (utilizando el mismo método que se muestra en el programa anterior), se procederá como sigue:

```
ZAP     DIVIDEND,WCALIF
ZAP     DIVISOR,WTOTAL
SRP     DIVIDEND,3,0
DP      DIVIDEND,DIVISOR
SRP     QUOTIENT,64-1,5
ZAP     PK2,QUOTIENT
MVC     OPCT,=X'40202120'
ED      OPCT,PK2
MVI     OPCT+L'OPCT,PERCENT
```

(En aras de la brevedad, he omitido el código para comprobar la división por cero.)

Observe el uso de la instrucción de la MVI y el uso del operador de longitud como un desplazamiento para mover un signo de porcentaje inmediatamente después de que el valor del porcentaje.

Una lógica similar se necesita al final del trabajo * (en Wrapup) para calcular por ciento del total. El resultado final es:

COGSWORTH INDUSTRIES
California's Contribution to Sales

Product	Nationwide Sales	California Sales	Percent of National
GIZMOS	90	20	22%
WIDGETS	37	15	41%
JUNQUE	73	25	34%
TOTALS	200	60	30%

El programa completo, COGS13B.MLC, sigue.

327 ⟨*FF* 327⟩≡ 329▷

```
        PRINT NOGEN
*****************************************************************
*           FILENAME: COGS13B.MLC                               *
*           AUTHOR : Bill Qualls                                *
*           SYSTEM : PC/370 R4.2                                *
*           REMARKS : Produce report for COGSWORTH INDUSTRIES *
*                     California's contribution to sales.*
*****************************************************************
        START 0
        REGS
```

```
BEGIN       BEGIN
            WTO       'COGS13B ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
            WTO       'COGS13B ... Normal end of program'
            RETURN (continued)
```

Recuerde, nunca es apropiado utilizar los porcentajes individuales (22, 41 y 34) para determinar un porcentaje total. Muchos programadores principiantes intento de llegar a la cifra final el promedio de los porcentajes individuales. Sin embargo, $(22 + 41 + 34)/3 = 32\,\%$, no el $30\,\%$. Por ciento del total es siempre un cálculo por separado basado en los totales de los datos originales, no los porcentajes individuales.

329 ⟨*FF* 327⟩+≡ ◁327

```
*****************************************************************
*        SETUP - Those things which happen one time only,               *
*                before any records are processed.                      *
*****************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         OI    INVENTRY+10,X'08'         PC/370 ONLY - Convert all
*                                        input from ASCII to EBCDIC
         OI    REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                        output from EBCDIC to ASCII
         OPEN INVENTRY
         OPEN REPORT
         BAL   R10,HDGS
         BAL   R10,READ
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU   *
         ST    R10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         PUT   REPORT,HD5
         PUT   REPORT,HD6
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.  *
*****************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         BAL   R10,FORMAT
         BAL   R10,WRITE
         BAL   R10,READ
         L     R10,SVPROC
```

```
         BR    R10
*****************************************************************
*        FORMAT - Format a single detail line.                  *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORM
         MVC   OREC,BLANKS
         MVC   ODESC,IDESC
         PACK WCALIF,ICALIF              Each product's sales must
         PACK WILL,IILL                    be packed so they can be
         PACK WUTAH,IUTAH                    added to total for this
         PACK WWISC,IWISC                     product...
         ZAP   WTOTAL,=P'0'              Initialize the total to zero
         AP    WTOTAL,WCALIF               and start adding...
         AP    WTOTAL,WILL
         AP    WTOTAL,WUTAH
         AP    WTOTAL,WWISC
         AP    TTOTAL,WTOTAL             Grand total nationwide
         AP    TCALIF,WCALIF             Grand total for Calif
         MVC   OTOTAL,=X'40202120'
         ED    OTOTAL,WTOTAL
         MVC   OCALIF,=X'40202120'
         ED    OCALIF,WCALIF
         ZAP   DIVIDEND,WCALIF
         ZAP   DIVISOR,WTOTAL
         SRP   DIVIDEND,3,0
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK2,QUOTIENT
         MVC   OPCT,=X'40202120'
         ED    OPCT,PK2
         MVI   OPCT+L'OPCT,PERCENT
         MVC   OCRLF,WCRLF                PC/370 only.
         L     R10,SVFORM
         BR    R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
READ     EQU   *
         ST    R10,SVREAD
         GET   INVENTRY,IREC              Read a single product record
         B     READX
ATEND    EQU   *
         MVI   EOFSW,C'Y'
READX    EQU   *
         L     R10,SVREAD
```

```
         BR    R10
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,OREC                Write report line
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         PUT   REPORT,HD6
         MVC   OREC,BLANKS
         MVC   ODESC(6),=C'TOTALS'
         MVC   OTOTAL,=X'40202120'
         ED    OTOTAL,TTOTAL
         MVC   OCALIF,=X'40202120'
         ED    OCALIF,TCALIF
         ZAP   DIVIDEND,TCALIF
         ZAP   DIVISOR,TTOTAL
         SRP   DIVIDEND,3,0
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK2,QUOTIENT
         MVC   OPCT,=X'40202120'
         ED    OPCT,PK2
         MVI   OPCT+L'OPCT,PERCENT
         MVC   OCRLF,WCRLF                PC/370 only.
         BAL   R10,WRITE
         CLOSE INVENTRY
         CLOSE REPORT
         WTO   'COGS13B ... Sales report on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
```

```
                    DDNAME='COGS.DAT'
          REPORT    DCB   LRECL=62,RECFM=F,MACRF=P,
                          DDNAME='REPORT.TXT'
 ************************************************************
 *        RETURN ADDRESSES                                  *
 ************************************************************
⟨search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP 109⟩
          ****************************************************************
          *        Miscellaneous field definitions                       *
          ****************************************************************
          WCRLF     DC    X'0D25'                    PC/370 ONLY - EBCDIC CR/LF
          EOFSW     DC    CL1'N'                     End of file? (Y/N)
          BLANKS    DC    CL62' '
          WCALIF    DC    PL2'0'                     Units sold in Calif
          WILL      DC    PL2'0'                     Units sold in Illinois
          WUTAH     DC    PL2'0'                     Units sold in Utah
          WWISC     DC    PL2'0'                     Units sold in Wisconsin
          WTOTAL    DC    PL2'0'                     Units sold in all states
          TCALIF    DC    PL2'0'                     Grand total for Calif
          TTOTAL    DC    PL2'0'                     Grand total nationwide
          PK2       DC    PL2'0'
          PERCENT EQU     C'%'
                    COPY DIVISION
                    COPY COGS
          ****************************************************************
          *        Output (line) definition                              *
          ****************************************************************
          OREC      DS    0CL62            1-62
          ODESC     DS    CL10             1-10 Product description
                    DS    CL7             11-17
          OTOTAL    DS    CL4             18-21 Units sold Nationwide
                    DS    CL9             22-30
          OCALIF    DS    CL4             31-34 Units sold in Calif
                    DS    CL8             35-42
          OPCT      DS    CL4             43-46 Percent sales from Calif
                    DS    CL14            47-60
          OCRLF     DS    CL2             61-62 PC/370 only - CR/LF
          ****************************************************************
          *        Headings definitions                                  *
          ****************************************************************
          HD1       DS    0CL62
                    DC    CL60'                      COGSWORTH INDUSTRIES           '
                    DC    XL2'0D25'
          HD2        DS     0CL62
                     DC     CL60'           California''s Contribution to Sales'
                     DC     XL2'0D25'
```

```
HD3      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'                    Nationwide            California '
         DC    CL20'Percent of'
         DC    XL2'0D25'
HD5      DS    0CL62
         DC    CL40' Product                 Sales                 Sales '
         DC    CL20' National '
         DC    XL2'0D25'
HD6      DS    0CL62
         DC    CL40'-----            -----            ----- '
         DC    CL20'-----'
         DC    XL2'0D25'
         END   BEGIN
```

13.5. Un ejemplo detallado

El siguiente ejemplo es más complejo y muestra el proceso de pensamiento necesario para hacer la multiplicación y la división de los problemas que aparecen en el mundo real ....

Supongamos que la actual tasa de impuesto de seguridad social (FICA) es 6,29 % y se aplica a los primeros $ 57.600 de ingreso solamente. Total de retenciones FICA no puede exceder $ 3,623.04. Datos de la nómina viene de una tarjeta de archivo de imagen e incluye los siguientes campos:

```
Input Fields
 IPPGR     DS     CL6     Current pay period Gross (9999V99)
 IYTDGR    DS     CL8     Year-to-date Gross (999999V99)
 IYTDFICA DS      CL7     Year-to-date FICA W/H (99999V99)
 Work Fields
 WFICA%    DC     PL3'0629'       FICA rate (V9999)
 WFICA$    DC     PL4'362304'     FICA dollar cap (99999V99)
```

Escriba el código de BAL necesario para producir los campos de salida siguiente. Definir los campos de trabajo adicional en caso necesario.

```
Output Fields
 OPPFICA DS       CL5     Current pay period FICA W/H (999V99)
 OYTDGR    DS     CL8     Year-to-date Gross (999999V99)
 OYTDFICA DS      CL7     Year-to-date FICA (99999V99)
```

13.6. Solución

Desde cada uno de los campos de entrada y de salida se divide en zonas decimal (sin envasar, como se indica, por definición, como CL), y desde cada uno será la entrada, o el resultado de, la aritmética, vamos a definir un campo correspondiente envasados de tamaño apropiado para cada uno:

```
WPPGR       DS     PL4      Current pay period Gross (9999V99)          - I
WYTDGR      DS     PL5      Year-to-date Gross (999999V99)              - I
WYTDFICA    DS     PL4      Year-to-date FICA W/H (99999V99)            - I
WPPFICA     DS     PL3      Current pay period FICA W/H (999V99)        - O
```

Let's paquete de cada uno de los campos de entrada:

```
PACK   WPPGR,IPPGR
PACK   WYTDGR,IYTDGR
PACK   WYTDFICA,IYTDFICA
```

es el más fácil de campo para determinar: sólo tiene que añadir WPPGR a WYTDGR. El campo de salida se divide en zonas OYTDGR decimal, por lo que la suma debe ser descomprimido y el signo eliminado:

```
AP      WYTDGR,WPPGR          Add current pay period to year-to-date
                UNPK    OYTDGR,WYTDGR Move new YTD gross to output
                MVZ     OYTDGR+L'OYTDGR-1(1),=X'F0'                  Remove sign
```

El importe FICA para el período de pago actual será el menor de (1) el período de pago actual veces la tasa bruta de FICA, o (2) la diferencia entre el límite monetario FICA y el Año-a-fecha de las retenciones FICA.

Para calcular el monto en primer lugar, hay que multiplicar un número con dos decimales (WPPGR) por otro número de cuatro cifras decimales (WFICA %), y luego redondear el resultado de nuevo a dos decimales. Estos campos se definen como PL4 y PL3 (arriba) por lo que utilizaremos el trabajo de campo de siete bytes para el producto a fin de evitar cualquier error de tamaño con el comando de multiplicar:

```
                ZAP     PK7,WPPGR             PK7 is V99
                MP      PK7,WFICA%             Now PK7 is V999999
                SRP     PK7,64-4,5            Now PK7 is back to V99, rounded
                ZAP     WPPFICA,PK7           This is pay period FICA amount
    *                                         unless it puts us over the cap
```

Vamos a llamar a la segunda cantidad (la diferencia entre el límite monetario FICA y el Año-a-fecha de las retenciones FICA) WMAXFICA, definido como:

```
    WMAXFICA DS         PL4
```

Cubierta para el actual período de pago FICA W / H, V99 Este campo se definió como PL4 por lo que podría celebrar WFICA $, que es necesaria para el cálculo. La tapa del FICA y las cantidades YTD son V99, por lo que sólo resta:

```
ZAP     WMAXFICA,WFICA$           Max this period is yearly max ...
                SP      WMAXFICA,WYTDFICA         less what has already been withheld
```

Ahora determinar la menor de estas dos cantidades, WPPFICA y WMAXFICA, como sigue:

```
                CP      WPPFICA,WMAXFICA    Over the cap?
                BL      SKIP                No - use percent of gross
                ZAP     WPPFICA,WMAXFICA    Yes - use what remains of the cap
    SKIP        EQU     *
                UNPK    OPPFICA,WPPFICA     Move pay period FICA to output
                MVZ     OPPFICA+L'OPPFICA-1(1),=X'F0'     Remove sign
```

Por último, la actualización de año hasta la fecha de las retenciones FICA:

```
AP      WYTDFICA,WPPFICA    Add pay period FICA to YTD FICA
                UNPK    OYTDFICA,WYTDFICA Move new YTD FICA amount to output
                MVZ     OYTDFICA+L'OYTDFICA-1(1),=X'F0'      Remove sign
```

Y hemos terminado.

13.7. Ejercicios

1. 1. Verdadero o falso. Dado A PL5'2000 DC, DC PL2'15 B, y C ... DC PL3'6 '

   a) MP A, B es válido, pero MP B, A no lo es. F T

   b) b. MP A, C es válido, pero MP C, A no lo es. F T

   c) MP C, B y C de SRP, 1,0 dar el mismo resultado en C. F T

   d) Si A y B representa el 20,00 a continuación representa el 1,5 MP A, B da una igualdad de F T X'000000030C que representan a 30.000.

   e) SRP A, 62,0 y SRP A ,64-2, 5 dan el mismo resultado en A. F T

   f) SRP B, 63,0 y SRP B ,64-1, 5 dan el mismo resultado en B. F T

   g) DP A, B da X'00133C003C A igual ' F T

   h) DP A, C da X'333C00002C A igual ' F T

   i) DP C = PL1'2 'C X'003C0C da igual " F T

   j) DP C = PL2'2 'no es válida. F T

   k) Después de DP A, = PL1'5 ', el resto es de +4 (1). F T

   l) Después de DP A, = PL2'5 ', el cociente es A (3). F T

   m) Después de DP A, = PL3'5 ', el cociente es A (2). F T

2. Te dan las definiciones siguientes campos:

```
IREG        DS    CL3     Input Regional total, 999V
   INAT        DS    CL4     Input National total, 9999V
   OPCT        DS    0CL8    Output Percent region to nation, BZZ9.99%
   ONBR        DS    CL7     Output Mask gets moved here - BZZ9.99
   OSIGN       DS    CL1     Output Percent sign gets moved here
   WREG        DS    PL2     Packed work field, Regional total
   WNAT        DS    PL3     Packed work field, National total
   DIVIDEND    DS    0PL16
   QUOTIENT    DS    PL8
   REMAINDR    DS    PL8
   DIVISOR     DS    PL8
```

   Vamos a dividir el total de la región por el total nacional que da el por ciento del total de la % BZZ9.99 forma. Uso de los campos dado sólo, escriba las instrucciones para:

   a) Mueva el total de aportaciones regionales (decimal zonificado) a su campo de trabajo de pic nic,

   b) Mueva el total de los aportes nacionales (decimal zonificado) a su campo de trabajo de pic nic,

c) Mueva el total regional envasados al dividendo,

d) Mueva el total nacional lleno hasta el divisor,

e) Si el divisor es igual a cero, cero a mover el campo de salida y continuar en la brecha,

f) De lo contrario el uso SRP para multiplicar el dividendo por 100000,

g) Dividir el dividendo por el divisor,

h) Ronda el cociente,

i) Mover la máscara de edición adecuada al campo de salida,

j) Editar el cociente en el campo de la producción, y

k) Mueva el signo de porcentaje para el campo de salida.

l) Escribir un programa para hacer todo lo anterior. Prueba PC/370 usando 's laboratorio.

3. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Escribir un programa que muestra el costo total de inventario a la mano de la pequeña ciudad de la tienda de hardware. Esta se define como la suma del coste de la (* cantidad de mano) para todos los instrumentos y envolturas. La salida debería ser por la OMC sólo: no hay ningún archivo de salida. Su mensaje debe aparecer como sigue:

```
Total cost of inventory on hand is BZZ,ZZ9.99.
```

(Nota: sería conveniente incluir un signo de dólar en la salida, pero PC/370 's aplicación de la orden de la OMC no permitirá que un signo de dólar. Esto se puede ver por sí mismo mediante el examen de la. PRN archivo de cualquier programa que utiliza una OMC y el control de la expansión de la macro de la OMC.)

4. Modificar el programa de la ficha cruz en Bolsas de la aritmética decimal, Ejercicio 10, para mostrar los porcentajes, así como los cargos. Su resultado debería aparecer como sigue:

```
          1         2         3         4
1234567890123456789012345678901234567890
         STUDENT STATISTICS
Status       Male     Female Total
--------    ------    ------ -----
Single      BZZ9      BZZ9   BZZ9
            BZZ9%     BZZ9%  BZZ9%
Married     BZZ9      BZZ9   BZZ9
            BZZ9%     BZZ9%  BZZ9%
--------    ------    ------ -----
Total       BZZ9      BZZ9   BZZ9
            BZZ9%     BZZ9%  BZZ9%
```

5. Un decano de la pequeña ciudad universitaria de la Comunidad gustaría comparar la calificación media otorgada por supuesto. Utilice la tabla de grado para producir un informe. A es un valor de 4 puntos, B es un valor de 3 puntos, una C vale 2 puntos, un D vale 1 punto, y un F vale 0 puntos. El archivo deberá ser ordenada por el ID del curso. Su informe debe aparecer como sigue:

```
          1         2         3         4
12345678901234567890123456789012345678901
       SMALL TOWN COMMUNITY COLLEGE
         Average Grade by Course
        Course ID        Count           GPA
        ---------        -----         -----
           XXXXX         BZZ9        BZ.99
           XXXXX         BZZ9        BZ.99
```

   Por ejemplo, MA107 se toma 3 veces, con los grados B, A y D. El PAM se calcula como sigue:

```
B   A   D
3 + 4 + 1 = 8 / 3 = 2.666 → round to → 2.67
```

6. Ejercicio Modificar 5 para incluir una calificación promedio de todos los cursos. Nota: esta cifra no es un 6. promedio de los promedios.

7. Modificar el ejercicio 5 para incluir la descripción del curso, así como el número de curso. Esto requerirá el uso de el archivo del campo y la congruencia lógica.

8. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Elaborar un informe con marcas por artículo. De marcado se define como (venta costo) / costo. Ronda en su caso. No mostrar las envolturas, que se indican por un precio de venta de cero. Su informe debe aparecer como sigue:

```
1          2           3           4           5           6
12345678901234567890123456789012345678901234567890123456789012345678901
                              SMALL TOWN HARDWARE STORE
                                      Markup Report
TID           Description                Cost        Sell       Markup
---     ----------------------         ------      ------     ------
XXX     XXXXXXXXXXXXXXXXXXXXX        BZZ9.99     BZZ9.99      ZZ9%
XXX     XXXXXXXXXXXXXXXXXXXXX        BZZ9.99     BZZ9.99      ZZ9%
XXX     XXXXXXXXXXXXXXXXXXXXX        BZZ9.99     BZZ9.99      ZZ9%
```

9. (Consulte la base de datos pequeña ciudad en la nómina más conjuntos de datos.)

a) Utilice el EMPL y tablas THISPP para determinar el salario bruto por empleado y en su conjunto para este período de pago. Los empleados se les paga por hora de tiempo y medio por hora más de cuarenta años. Empleados asalariados se les paga una cantidad fija, independientemente del número de horas trabajadas. Ambos archivos deberán ser ordenados y emparejado en el número de empleado. Su resultado debería aparecer como sigue:

```
1         2          3          4
          12345678901234567890123456789012345678901234567890
          PAYROLL REPORT FOR WEEK ENDING 01/09/93
          ENUM Type      Hours       Rate         Gross
          ---- ----     -------     ------     ---------
          XXX       X    BZZ9.99    BZZ9.99       BZZ9.99
          XXX       X    BZZ9.99    BZZ9.99       BZZ9.99
          ---- ----     -------     ------     ---------
          TOTALS         BZZ9.99               BZZ,ZZ9.99
```

b) Modificar el programa de la parte (a) para anexar los resultados de este período de pago (01/09/93) para el archivo histórico. (Para ello será necesario crear un archivo nuevo: lo llaman HISTORY.NEW.)

Capítulo 14

Binary Arithmetic

Objetivos Al término de este capítulo usted será capaz de:

- Explique qué se entiende por la palabra doble, fullword, y la alineación de la frontera media palabra,
- Dado LOC un campo indican si es palabra doble, fullword, y / o media palabra alineados,
- Utilice las siguientes instrucciones binarias: L, LH, LR, A, AH, AR, S, SH, SR, C, CH, y CR,
- Usar la Oficina de Turismo de la instrucción para convertir una serie de pic nic a binario, y la
- Utilice la instrucción de las ECV para convertir un número binario de pic nic.

Introducción Los equipos System/370 sólo puede hacer dos tipos de aritmética: embalado y binarios. Hemos ya miró la aritmética envasados en detalle. En este capítulo hablaremos de la aritmética binaria. La aritmética binaria es también conocido como registro de la aritmética, ya que cada operación aritmética binaria participarán al menos un registro. Hay varias razones por las que habría que saber binario la aritmética, además de la aritmética embalado:

- El archivo que está leyendo pueden haber sido creados con campos binarios.
- Aritmética binaria es necesaria en el tratamiento de registros de longitud variable.
- En algunos casos, los números de ahorro en forma binaria en lugar de en forma condensada puede dar como resultado menos

- almacenamiento. Si los datos se transmiten a través de un sistema de comunicaciones, está el adicional de beneficio de la reducción en la transmisión de datos. Procesamiento de la tabla en el BAL (el tema de nuestro próximo capítulo) requiere la aritmética binaria.

* * * * * * * * * * * * * * * * * * * *

Recuerde que las computadoras se basan en el sistema de numeración binario, y que se utiliza el formato hexadecimal sistema de numeración como una abreviatura conveniente de los binarios. Cada byte de memoria se ha asociado con una dirección, que podemos mostrar en formato hexadecimal. Como con muchas cosas en la informática, estos lugares están numerados dirección respecto a cero, es decir, empiezan en cero en lugar de uno. Así que la memoria puede ser representados como sigue:

```
000000                          00000F
       (4 bytes each column)
FFFFF0                          FFFFFF
```

Tenga en cuenta que cada "fila" en la figura es octetos dieciséis años en la longitud (la más a la derecha, o de orden inferior, de dos dígitos de las direcciones que van de 0 a F inclusive.) Memoria se divide en "palabras", cada uno de ellos es de cuatro bytes de longitud.

La primera palabra ocupa octetos 000000-000003 inclusive, la segunda palabra ocupa octetos 000004-000007 inclusive, la tercera palabra ocupa octetos 000008-00000b inclusive. Esto continúa hasta e incluyendo la última palabra que ocupa octetos FFFFFC-FFFFFF inclusive.

Recuerde que en un debate anterior que cada registro también ocupa cuatro bytes, y que es el Tamaño de registro que determina el tamaño de palabra de un ordenador.

Si el dígito más a la derecha de la dirección del primer byte de un campo es 0, 4, 8 o C, entonces decimos que el campo de es "fullword alineados", o "alineado en un límite de fullword". En el siguiente ejemplo, los campos A y el D son fullword alineados. ¿Por qué?

```
LOC
020104 A DS PL3
020107 B DS CL2
020109 C DS CL3
02010C D DS CL4
```

Utilizamos una especificación del tipo de campo de la F para definir una fullword. Por ejemplo:

```
FIVETHOU DC F'5000'
```

Un campo definido con una especificación del tipo de F se ve obligado a un límite de fullword. Esto puede ser una muy consideración importante en la definición de diseños de registro. Considere el siguiente ejemplo. ¿Puede usted suministro de los campos que faltan LOC?

```
020117 INREC   DS 0CL80
______ INZIP   DS PL3
______ INCODE  DS CL1
______ INAMT   DS F
______ INJUNK  DS CL6
______ INLIMIT DS F
```

(Usted debe terminar con INLIMIT a partir de las LOC = 020128, porque 02011B, 020126 y 020127 se "omitido".)

Puedo resolver el problema de saltar 02011B obligando a todo el registro para comenzar en un fullword frontera. Esto se hace con una DS 0F como se muestra. Una vez más, se puede suministrar la LOC falta?

```
020118         DS 0F
______ INREC   DS 0CL80
______ INZIP   DS PL3
______ INCODE  DS CL1
______ INAMT   DS F
______ INJUNK  DS CL6
______ INLIMIT DS F
```

(Una vez más, usted debe terminar con INLIMIT a partir de las LOC = 020128 por 020126, y 020127 se todavía "omitido".)

Teniendo en cuenta estos campos se definen en el orden indicado, no hay manera de que pueda vigor INAMT y INLIMIT a un límite fullword. Cuando se utiliza un diseño de registro con campos definidos como fullwords, es común la práctica a:

- Utilice DS 0F para forzar el comienzo del registro a un límite fullword, y
- Asegúrese de que todos los campos definidos como de tipo F comenzar en la columna (n * 4) +1, es decir, las columnas 1, 5, 9, 13, ...

Esta segunda regla se puede lograr mediante la definición de un registro de tal manera que se definen todos los campos de tipo F. antes de que otros campos. Esta es una práctica común en algunas tiendas. De hecho, algunas instalaciones se ir tan lejos como para decir que todos los diseños de registro se define de modo que su longitud total es un múltiplo de cuatro, incluso si esto significa "relleno" el registro con unos pocos bytes de "relleno". Por supuesto, estos son consideraciones hechas en el momento del diseño de registro está diseñado: se trata de cuestiones de diseño que se deberían haber sido contestadas antes de que el programa llegue a usted!

Dado un fullword es de cuatro bytes de longitud, se deduce que una palabra doble es de ocho bytes de longitud. Usamos una especificación de tipo de campo de la D para definir una palabra doble. Por ejemplo:

```
BILLION DC D'1000000000'
```

Si el dígito más a la derecha de la dirección del primer byte de un campo es de 0 a 8 entonces decimos que es "palabra doble alineados" o "alineado en un límite de palabra doble". Un campo definido con un tipo de la especificación de la D se ve obligado a un límite de palabra doble.

Del mismo modo, una media palabra es de dos bytes de longitud. Utilizamos una especificación del tipo de campo de la H para definir un media palabra. Por ejemplo:

```
MINUSONE DC H'-1'
```

Si el dígito más a la derecha de la dirección del primer byte de un campo es 0, 2, 4, 6, 8, A, C, E o decimos que es "media palabra alineados" o "alineado en un límite de media palabra". Un campo definido con un tipo de la especificación de la H se ve obligado a un límite de media palabra.

La siguiente tabla puede ayudar a ilustrar la "jerarquía" de doublewords, fullwords, medias palabras, y los bytes.

```
                          DBL
        FULL1                          FULL2
 HALF1          HALF2          HALF3          HALF4
BYTE1  BYTE2  BYTE3  BYTE4  BYTE5  BYTE6  BYTE7  BYTE8
```

¿Cómo podrían estos ámbitos se definirán en un programa?

Solución

```
DBL   DS 0D  1-8
FULL1 DS 0F  1-4
HALF1 DS 0H  1-2
BYTE1 DS CL1 1-1
BYTE2 DS CL1 2-2
HALF2 DS 0H  3-4
BYTE3 DS CL1 3-3
BYTE4 DS CL1 4-4
FULL2 DS 0F  5-8
HALF3 DS 0H  5-6
BYTE5 DS CL1 5-5
BYTE6 DS CL1 6-6
HALF4 DS 0H  7-8
BYTE7 DS CL1 7-7
BYTE8 DS CL1 8-8
```

intenta 1. El siguiente registro se define como 24 bytes, pero en realidad se ocupan más que eso. Suministrar LOC los desaparecidos y determinar el número de bytes ocupados por este registro, dado que el COL del primer byte es 020100. (Usted debe terminar con LOC = 02011C para BIG7.) Luego reordenar la campos en el registro de manera que el registro, de hecho, ocupan sólo 24 bytes. (Usted debe terminar con LOC = 020117 para BIG7.)

```
LOC
020100 BIGMESS DS 0CL24
______ BIG1    DS CL5   1-5
______ BIG2    DS D     6-13
______ BIG3    DS CL3   14-16
______ BIG4    DS H     17-18
______ BIG5    DS CL1   19
______ BIG6    DS F     20-23
______ BIG7    DS CL1   24
```

* * * * * * * * * * * * * * * * * * * *

Habiendo examinado los tres tamaños de los campos binarios - palabra doble, fullword, y halfword - ahora vistazo a la "capacidad" de estos campos, es decir, el rango de valores que cada uno puede contener. Empezamos con la media palabra, ya que es la más fácil. Sabemos que una media palabra ocupa dos bytes, y sabemos que de cada byte puede ser representado por dos dígitos hexadecimales, cada uno de los cuales representa cuatro bits. Recuerde que en el nuestro análisis de números empaquetados, el más a la derecha de dos dígitos hexadecimales representan el signo (F, D, o C). Evidentemente, algún mecanismo similar se utiliza para indicar el signo de un número binario.

La clave para entender los números binarios es reconocer que se utiliza más a la izquierda (más alto) poco para representar el signo: este bit estará apagado (0) para los números positivos y en (1) para los números negativos. (Nota: Considerando que un número puede ser envasados sin firma, no hay números binarios sin signo.) Por lo tanto, el rango de valores para una media palabra positiva es:

```
0000 0000 0000        0000 Binary                    = 0 10          0    0    0    0
```

a través de ...

```
1111 1111 1111                              0111                 Binary = 32767 10
```

El rango de valores para una media palabra negativa es:

```
1111 1111 1111                              1111                 Binario = -1 10
```

El hecho de que X'FFFF '= -1 es contrario a la intuición. Pero pensar de esta manera ... si añadimos X'0001 'para X'FFFF 'tenemos X'10000', que cuando se trunca a dos bytes es X'0000 '. Así, vemos que -1 + 1 = 0. Por lo tanto, X'FFFF 'debe ser igual a -1.

Bolsas de campo de dos bytes pueden variar en el valor de -999 a 999 solamente. Pero, como hemos visto, un halfword puede variar en el valor de -32768 a 32767 inclusive. Ahora supongamos que queremos almacenar un valor que

nunca saldrá de ese rango. Para almacenar -32768 o 32.767 en forma de pic nic que requieren de tres bytes de almacenamiento (X'32768D y X'32767C '). Pero para almacenar estos mismos valores en un halfword requiere sólo dos bytes de almacenamiento, un ahorro del 33 %.

Sabemos que un fullword ocupa cuatro bytes. Siguiendo el mismo patrón que el anterior, es decir, utilizando el bit más a la izquierda para indicar que el signo, vemos que el rango de valores para un fullword positivo es:

```
0000 0000 0000 0000 0000 0000 0000 0000          Binario = 0 10                    0 0
```

a través de ...

```
1111 1111 1111 1111 1111 1111 1111                 0111          Binario = 2.147.483.6
```

El rango de valores para un fullword negativo es:

```
1111 1111 1111 1111 1111 1111 1111                 1111          Binario = -1 10
```

a través de ...

```
0000 0000 0000 0000 0000 0000 0000                 1000          Binario = -2147483648
```

Tenga en cuenta que se necesitarían seis bytes para almacenar este mismo número en formato compacto. (Pero, por supuesto, los seis bytes podría almacenar un número de envasado en la gama ??? 99999999999).

El rango de valores para una palabra doble se -9.223.372.036.854.775.808 a +9.223.372.036.854.775.807 (Lo suficientemente grande para la mayoría de las empresas!)

14.1. Comparación de dos campos binarios

Cualquier operación aritmética binaria (incluyendo una comparación) necesitará por lo menos un registro. Supongamos que A y B son cada fullwords. A tiene un valor de 5 y B tiene un valor de -5. Pueden ser se muestra (en hexadecimal) como:

```
00 00 00 05 FF FF FF FB              A = B =
```

No podemos utilizar CLC para comparar los campos:

```
CLC A, B
```

daría lugar a una baja, lo que claramente no es el caso. En su lugar, utilice la instrucción de carga (L) para el puesto de cada fullword en un registro separado, y la instrucción de comparación registro (CR) para comparar los dos registros:

```
L  R3,A
L  R4,B
CR R3,R4
```

Alternativamente, se podría cargar uno de estos campos en un registro y luego usar la comparación enseñanza (C) para comparar ese registro a la fullword otros:

```
L R3,A or... L R4,B
C R3,B       C R4,A
```

¿Qué pasaría si se comparan dos medias palabras? Por ejemplo, supongamos que es un media palabra con un valor de C 2 y D es una media palabra con un valor de -2:

```
   00 02    FF FE C=       D=
```

Podemos utilizar la instrucción media palabra de carga (LH) para colocar cada media palabra en un registro separado, y la instrucción de comparación registro (CR) para comparar los dos registros:

```
LH R3,C
LH R4,D
CR R3,R4
```

Alternativamente, se podría cargar uno de estos campos en un registro y luego usar la comparación enseñanza media palabra (CH) para comparar ese registro a la media palabra de otros:

```
LH R3,C or... LH R4,D
CH R3,D       CH R4,C
```

Tenga en cuenta que cuando se utiliza la instrucción de la LH, el signo se conserva, es decir, el bit de signo de la media palabra (poco más a la izquierda) se propagó por el resto del registro. Esto se puede mostrar como:

```
1111 1111 1111 1110                                D = D = -210 Binario
```

... después de que se convierte en ...

```
LH R4, D                    1111 1111 1111 1111 1111 1111 1110                1111
```

... no ...

```
0000 0000 0000 1111 1111 1111 1110                0000        R4 binario = 32766 10
```

Tenga mucho cuidado que se utiliza para cargar una L fullword y LH para cargar una media palabra. Si no lo hace se no causa una excepción decimal (S0C7), pero sin duda dará resultados erróneos. Por ejemplo, si C y D se ha definido que el anterior, y en ese orden, y si hubiera uso L (en lugar de LH) para poner en C registro 3, que registró en ese momento debería contener:

```
0000 0000 0000 0010 1111 1111 1111 1110        Binary R3 = 196606 10
```

Cabe señalar que cualquier combinación de bits es un número binario válido. Por lo tanto, a diferencia de Bolsas de aritmética decimal, la aritmética binaria nunca se produce un abend excepción de datos (S0C7). Esto no significa que la matemática binaria deben tener prioridad sobre las matemáticas decimal empaquetado. Como acabamos de muestra, todavía puede cometer un error al tratar campos binarios de forma incorrecta. El resultado es errores sin previo aviso. Al menos las operaciones decimal empaquetado advertir (por desagradable que la alerta puede ser) que se han utilizado los datos no numéricos.

Para completar, podemos mencionar aquí que el registro de instrucciones de carga (LR) copia el valor de una registro (el segundo operando) a otro registro (el primer operando).

intenta 2. Teniendo en cuenta completa es una fullword y la mitad es una media palabra, muestran tres formas diferentes para comparar COMPLETO y medio. En cada caso, rama en la etiqueta bajo si el valor de FULL es menor que el valor de la La mitad.

14.2. Mover un registro a una Fullword o halfword

En el capítulo cuatro se vio que lo contrario de la instrucción de carga (L) es la instrucción de la tienda (ST): de las copias de la tienda de instrucción el valor de un registro a una fullword.

Del mismo modo, lo contrario de la enseñanza media palabra de carga (LH) es la instrucción media palabra de la tienda (STH): la enseñanza media palabra tienda se copia el valor de un registro a una media palabra, el mantenimiento de del itegrity del signo, sino también truncar si es necesario y sin previo aviso.

¿Qué debe hacer si usted sabe el valor de una fullword cabrá en una media palabra y usted quiere copia que fullword a una media palabra? Por ejemplo, dada:

```
FULL DS F
HALF DS H
```

... usted puede tener la tentación de hacer lo siguiente:

```
MVC HALF,FULL+2
```

... pero no lo hagas! Usted debe pasar por un registro, de lo contrario poner en peligro la integridad de la signo. La solución correcta es:

```
L   R5,FULL
STH R5,HALF
```

intenta 3. Escriba el código necesario para mover el valor de la mitad a pleno.

4. Escriba el código necesario para mover el mayor de mitad y la totalidad de Max, donde Max se define como la un fullword.

14.2.1. Además Binario

Un registro es siempre el campo de recepción de cualquier instrucción Además binario. Primero, utilice la L, LH, o LR instrucciones para colocar un operando en un registro. A continuación, utilice el complemento (A), agregue media palabra (AH), o Añadir registro (AR) la instrucción en función de si el segundo operando es fullword, media palabra, o registro respectivamente. Por ejemplo, dada:

```
A DS F
B DS F
C DS H
D DS F
```

... para añadir el que D, podemos código:

```
L  R3,A
A  R3,B
AH R3,C
ST R3,D
```

Otro ejemplo ... Dado A es una fullword, añadir 1 a A. Hay muchas maneras de que podamos hacer esto, entre ellos:

```
          or...
L  R3,A           L  R3,A
A  R3,=F'1'       AH R3,=H'1'
ST R3,A           ST R3,A
```

Una vez más, no hay manera de añadir a un número binario sin pasar por un registro. Dicho de otra Así, todas las instrucciones matemáticas binarias son de tipo RR o RX sólo: no hay ningún tipo de matemáticas binarias SS instrucciones.

intenta 5. Habida cuenta de F1 y F2 son fullwords, y H1 y H2 son medias palabras. Escriba el código necesario para poner el menor de (F1 + F2) y (H1 + H2) en el registro 7. Su código no debe cambiar los valores de la F1, F2, H1 y H2. No definen los campos de trabajo de otros. Tendrá dos registros.
* *

14.3. Binario Resta

Para la resta binaria, como con la suma binaria, en primer lugar utilizar el L, LH, o la instrucción LR colocar una operando en un registro. A continuación, utilice el sustrato (S), restar media palabra (SH), o restar registro (SR) instrucción en función de si el segundo operando es fullword, media palabra, o registro, respectivamente. Para ejemplo, dado:

```
A DS F
B DS F
C DS H
```

para calcular A = A-(B + C), en primer lugar reconocer que esto es equivalente a A = ABC. A continuación, el código:

```
L  R3,A
S  R3,B
SH R3,C
ST R3,A
```

intenta 6. Habida cuenta de F1 y F2 son fullwords, y H1 y H2 son medias palabras. Escriba el código necesario para calcular F1 = (F1-F2) + (H1-H2).

14.4. La conversión de Bolsas a Binary (CVB)

Suponga que los datos de entrada es de carácter (decimal zonas) o en forma de pic nic. ¿Cómo se puede convertir estos números a su equivalente binario? Esto se hace con la instrucción de convertir a binario (CVB). Esta instrucción se convierte un número envasados almacenados en una palabra doble, en un número binario almacenados en un registro. Por ejemplo, dada:

```
IAMOUNT DS CL4 Input amount, unpacked, 99V99
OAMOUNT DS H   Output amount, binary, 99V99
DBLWORD DS D   Doubleword work area
```

Tenemos el siguiente código:

```
PACK DBLWORD,IAMOUNT Must be packed to use CVB
CVB  R3,DBLWORD      Binary equivalent in R3
STH  R3,OAMOUNT      Binary equivalent to output
```

Tenga en cuenta que el segundo operando de la CVB debe ser un número válido de pic nic. Si no es así, entonces usted Abend obtener una excepción de datos. Además, este número debe ocupar Bolsas de los ocho bytes de la palabra doble.

Puede parecer un poco extraño que un número de relleno se están almacenando en una palabra doble. Sin embargo, podemos pensar en una palabra doble en este caso como ocho bytes que resultan ser palabra doble alineados. Si DBLWORD se había definido como DC D'0 ', entonces no comienza como un número válido de pic nic, pero el Instrucción PACK sustituirá el cero binario con un número de pic nic.

14.5. Conversión de binario a Bolsas (ECV)

Lo contrario de la Oficina de Turismo de ECV es: convertir a decimal. Esta instrucción se convierte el contenido de registro en un número embalado y almacenar ese número embalado en una palabra doble. La instrucción de las ECV, como la instrucción de ST, es una de las pocas instrucciones en el segundo operando especifica el campo de recepción (recordar que en la mayoría de los casos, el segundo operando es el campo de origen y de la primera operando es el campo de recepción.)

Usted tendrá que usar la instrucción para convertir a las enfermedades cardiovasculares (con embalaje) decimales si no por otra razón de imprimir el número. No hay un equivalente a la instrucción ED para números binarios: un binario número debe ser convertido a forma de pic nic antes de editarlo. Considere el siguiente ejemplo. Teniendo en cuenta:

```
GROSS    DS F    Gross sales, 99999V99
SALESTAX DS F    Sales tax, 9999V99
NET      DS CL10 Net = Gross + Tax, BZZ,ZZ9.99
DBLWORD  DS D    Work field
```

para añadir a continuación ImpuestoVenta a BRUTO dando NET, con formato, que el código:

```
L   R4,GROSS       Gross sales in R4
A   R4,SALESTAX    Add sales tax
CVD R4,DBLWORD     Convert to packed for printing
MVC NET,=X'4020206B2021204B2020'   BZZ,ZZ9.99
ED  NET,DBLWORD+4 Seven digits in NET so edit the
                   last four bytes of DBLWORD only
```

intenta 7. Dado CINCO H'5 DC, tres DC H'3 ', y RESULTADO DC X'40202120', escribir un programa corto, que restará Tres de CINCO dando RESULTADO. Uso de la OMC para mostrar RESULTADO. Definir los campos de otra índole que es necesario, pero no todos los campos de la aritmética en binario, es decir, el uso embalados necesaria para la ED comando.

14.6. Muestra del Programa - Datos Dindón Convertir a binario

Para ilustrar aún más los conceptos que han sido presentado anteriormente, veremos a los dos Inventario de datos, ejemplos de programación. Recordemos el Inventario de W ASCII / números Las discrepancias informe para las Industrias Dindón, COGS.DAT en Zonalizada Decimal presentado en un capítulo anterior. El formato de archivo de inventario. Se COGS.DAT. Los campos numéricos en ese archivo se almacenados en formato decimal en zonas. Vamos a escribir dos Convertir datos programas. El primer programa va a crear un nuevo archivo COGS14A.MLC a formato binario con todos los campos numéricos almacenados en medias palabras. El segundo programa leerá este nuevo archivo y producen las discrepancias informe de inventario. Esto Inventario de datos, se muestra en el diagrama de flujo del sistema hacia la derecha. W EBCDIC / números COGS.BIN en formato binario.

Veamos primero en el programa de conversión, COGS14A.MLC.

Producir Inventario Discrepancias COGS14B.MLC Informe Todos los campos numéricos se almacenan como medias palabras. ?? a partir de datos binarios. Recordemos, por tanto, de nuestra discusión anterior que es esencial que nosotros le garantizamos media palabra Las discrepancias de inventario alineación. El diseño de registro de salida se muestra informe.txt Informe, ASCII, Imprimir la imagen. por debajo de. Tenga en cuenta que las medias palabras se puede definir después de la descripción ya que la longitud de la descripción (10) es un múltiplo de la longitud de una media palabra (2).

```
        DS 0H           Force halfword alignment
OREC    DS 0CL28  1-28 Inventory record
ODESC   DS CL10   1-10 Product description
OCALIF DS H     11-12 Units sold in Calif
OILL    DS H     13-14 Units sold in Illinois
OUTAH   DS H     15-16 Units sold in Utah
OWISC   DS H     17-18 Units sold in Wisconsin
OBEGIN DS H     19-20 Beginning inventory
OPURCH DS H     21-22 Purchases throughout year
OQOH    DS H     23-24 Actual quantity on hand
OCOST   DS H     25-26 Cost (each) 99V99
OSELL   DS H     27-28 Sell for (each) 99V99
```

Tenga en cuenta que no hay CR / LF al final del registro. El CR / LF no es necesario: hemos utilizado en nuestros programas anteriores para que pudiéramos utilizar el comando de DOS, "para ver nuestros datos. Desde que este disco contiene los números binarios, el único campo que puede "ver" es la descripción, por lo que se omite el CR / LF.

- Necesitamos un DCB para ambos archivos. Mostramos aquí la DCB para el archivo de salida. No hay nada ?? inusual aquí. Por supuesto, la LRECL

debe coincidir con la duración se muestra en el diseño de registro de salida arriba (28).

```
BINARY DCB LRECL=28,RECFM=F,MACRF=P,
           DDNAME='COGS.BIN'
```

- Dado que nuestros datos contiene los datos binarios, no podemos utilizar la conversión de EBCDIC a ASCII! Eso ?? tipo de conversión, simplemente no funcionará, ya que no es un uno-a-uno entre los códigos ASCII y EBCDIC. Así que vamos a omitir la instrucción de OI que se utilizan para ya antes de la instrucción OPEN.

```
  OI   INVENTRY+10,X'08' PC/370 ONLY - Convert all
*                        input from ASCII to EBCDIC
  OPEN INVENTRY
  OPEN BINARY            NOTE: Output in EBCDIC
```

  Por supuesto, para leer este archivo nuevo, el programador debe saber que ya está EBCDIC y es sin CR / LF.

- Dentro de la rutina de formato, cada campo numérico serán empacadas, convertido a binario, y se mudó ?? al campo de salida correspondiente. El código para hacerlo es la siguiente:

```
PACK DBL,ICALIF Convert CALIF to binary
CVB  R3,DBL
STH  R3,OCALIF
PACK DBL,IILL   Convert ILL to binary
CVB  R3,DBL
STH  R3,OILL
PACK DBL,IUTAH  Convert UTAH to binary
CVB  R3,DBL
STH  R3,OUTAH
PACK DBL,IWISC  Convert WISC to binary
CVB  R3,DBL
STH  R3,OWISC
PACK DBL,IBEGIN Convert BEGIN to binary
CVB  R3,DBL
STH  R3,OBEGIN
PACK DBL,IPURCH Convert PURCH to binary
CVB  R3,DBL
STH  R3,OPURCH
PACK DBL,IQOH   Convert QOH to binary
CVB  R3,DBL
STH  R3,OQOH
PACK DBL,ICOST  Convert COST to binary
CVB  R3,DBL
```

```
        STH  R3,OCOST
        PACK DBL,ISELL  Convert SELL to binary
        CVB  R3,DB
        L STH  R3,OSELL
```

El programa completo, COGS14A.MLC, sigue.

353 ⟨*COGS14A.MLC* 353⟩≡

```
                PRINT NOGEN
          ****************************************************************
          *        FILENAME: COGS14A.MLC                                 *
          *        AUTHOR : Bill Qualls                                  *
          *        SYSTEM : PC/370 R4.2                                  *
          *        REMARKS : Create binary data file using COGS.DAT      *
          ****************************************************************
                  START 0
                  REGS
          BEGIN   BEGIN
                  WTO    'COGS14A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
                  WTO    'COGS14A ... Normal end of program'
                  RETURN
          ****************************************************************
          *        SETUP - Those things which happen one time only,      *
          *                 before any records are processed.            *
          ****************************************************************
          SETUP   EQU    *
                  ST     R10,SVSETUP
                  OI     INVENTRY+10,X'08'          PC/370 ONLY - Convert all
          *                                         input from ASCII to EBCDIC
                  OPEN INVENTRY
                  OPEN BINARY                       NOTE: Output in EBCDIC
                  BAL    R10,READ
                  L      R10,SVSETUP
                  BR     R10
          ****************************************************************
          *        PROCESS - Those things which happen once per record.  *
          ****************************************************************
          PROCESS EQU    *
                  ST     R10,SVPROC
                  BAL    R10,FORMAT
                  BAL    R10,WRITE
                  BAL    R10,READ
                  L      R10,SVPROC
                  BR     R10
****************************************************************
*        READ - Read a record.                                 *
```

```
*****************************************************************
⟨search..., read INVENTRY into IREC, EOFSW, SVREAD return 314⟩
          ***************************************************************
          *       FORMAT - Format a single detail line.                 *
          ***************************************************************
          FORMAT   EQU   *
                   ST    R10,SVFORM
                   MVC   ODESC,IDESC
                   PACK DBL,ICALIF                 Convert CALIF to binary
                   CVB   R3,DBL
                   STH   R3,OCALIF
                   PACK DBL,IILL                   Convert ILL to binary
                   CVB   R3,DBL
                   STH   R3,OILL
                   PACK DBL,IUTAH                  Convert UTAH to binary
                   CVB   R3,DBL
                   STH   R3,OUTAH
                   PACK DBL,IWISC                  Convert WISC to binary
                   CVB   R3,DBL
                   STH   R3,OWISC
                   PACK DBL,IBEGIN                 Convert BEGIN to binary
                   CVB   R3,DBL
                   STH   R3,OBEGIN
                   PACK DBL,IPURCH                 Convert PURCH to binary
                   CVB   R3,DBL
                   STH   R3,OPURCH
                   PACK DBL,IQOH                   Convert QOH to binary
                   CVB   R3,DBL
                   STH   R3,OQOH
                   PACK DBL,ICOST                  Convert COST to binary
                   CVB   R3,DBL
                   STH   R3,OCOST
                   PACK DBL,ISELL                  Convert SELL to binary
                   CVB   R3,DBL
                   STH   R3,OSELL
                   L     R10,SVFORM
                   BR    R10
          ***************************************************************
          *       WRITE - Write a single output record.                                 *
          ***************************************************************
          WRITE    EQU   *
                   ST    R10,SVWRITE
                   PUT   BINARY,OREC
                   AP    #OUT,=P'1'
                   L     R10,SVWRITE
                   BR    R10
```

```
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                  after all records have been processed.       *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE INVENTRY
         CLOSE BINARY
         WTO   'COGS14A ... Binary file COGS.BIN created.'
         ED    MSG#OUT,#OUT
         WTO   MSG
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=41,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.DAT'
BINARY   DCB   LRECL=28,RECFM=F,MACRF=P,
               DDNAME='COGS.BIN'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP DC     F'0'              SETUP
SVPROC   DC    F'0'              PROCESS
SVREAD   DC    F'0'              READ
SVFORM   DC    F'0'              FORMAT
SVWRITE DC     F'0'              WRITE
SVWRAP   DC    F'0'              WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'           PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'            End of file? (Y/N)
#OUT     DC    PL2'0'            Count of records written
DBL      DC    D'0'              To convert packed to binary
         COPY COGS
*****************************************************************
*        Output record definition                               *
*****************************************************************
         DS    0H                Force halfword alignment
OREC     DS    0CL28         1-28 Inventory record
```

```
ODESC    DS    CL10          1-10 Product description
OCALIF   DS    H            11-12 Units sold in Calif
OILL     DS    H            13-14 Units sold in Illinois
OUTAH    DS    H            15-16 Units sold in Utah
OWISC    DS    H            17-18 Units sold in Wisconsin
OBEGIN   DS    H            19-20 Beginning inventory
OPURCH   DS    H            21-22 Purchases throughout year
OQOH     DS    H            23-24 Actual quantity on hand
OCOST    DS    H            25-26 Cost (each) 99V99
OSELL    DS    H            27-28 Sell for (each) 99V99
*****************************************************************
*        Output message (count of records written)              *
*****************************************************************
MSG      DS    0CL32
         DC    CL11'COGS14A ...'
MSG#OUT DC     XL4'40202120'
         DC    CL17' records written.'
         END   BEGIN
```

14.7. Muestra del Programa - Discrepancias Informe de Inventario del archivo binario

Veamos ahora el segundo programa, COGS14B.MLC. Este programa se presenta el inventario Las discrepancias informe del nuevo archivo (binario). (Nota: este programa es una modificación de COGS9B.MLC, que produjo el mismo informe de la coma decimal por zonas de archivo.)

- El registro de entrada contiene medias palabras, por lo que debe ser halfword alineados al igual que cuando la ?? El registro fue creado.

```
     DS 0H         Force halfword alignment
IREC DS 0CL28 1-28 Inventory record
```

  La LRECL del archivo de entrada DCB debe coincidir con la duración se muestra en el diseño de registro de entrada (28).

```
INVENTRY DCB LRECL=28,RECFM=F,MACRF=G,EODAD=ATEND,
             DDNAME='COGS.BIN'
```

  El archivo de entrada se creó en forma EBCDIC, por lo que omite la instrucción de OI que estamos acostumbrados ?? a ver antes de la instrucción OPEN.

```
  OI   REPORT+10,X'08' PC/370 ONLY - Convert all
*                      output from EBCDIC to ASCII
```

```
         OPEN INVENTRY        NOTE: Input in EBCDIC
         OPEN REPORT
```

- Todos los cálculos se hacen en binario. Vea la sección de formato. Usamos cuatro registros de estos ?? cálculos. Podríamos haber utilizado los registros de menos si hubiéramos estado dispuestos a almacenar algunos los resultados intermedios.

  Debería ser evidente al examinar el código en la sección de formato que el código podría convertirse en muy confuso con bastante rapidez. El uso liberal de comentarios significativos es esencial!

El programa completo, COGS14B.MLC, sigue.

357 ⟨*COGS14B.MLC* 357⟩≡

```
                PRINT NOGEN
*****************************************************************
*        FILENAME: COGS14B.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce report for COGSWORTH INDUSTRIES      *
*                    showing inventory discrepancies.           *
*                    Modify COGS9B.MLC to use binary input.     *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'COGS14B ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'COGS14B ... Normal end of program'
         RETURN
*************************************************************
*        SETUP - Those things which happen one time only,                *
*                before any records are processed.                       *
*************************************************************
SETUP    EQU   *
         ST    R10,SVSETUP
         OI    REPORT+10,X'08'          PC/370 ONLY - Convert all
*                                       output from EBCDIC to ASCII
         OPEN INVENTRY                  NOTE: Input in EBCDIC
         OPEN REPORT
         BAL   R10,HDGS
         BAL   R10,READ
         L     R10,SVSETUP
         BR    R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
```

```
HDGS     EQU   *
         ST    R10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         PUT   REPORT,HD5
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record. *
*****************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         BAL   R10,FORMAT
         BAL   R10,WRITE
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        READ - Read a record.                                  *
*****************************************************************
⟨search..., read INVENTRY into IREC, EOFSW, SVREAD return 314⟩
         *****************************************************************
         *        FORMAT - Format a single detail line.                                  *
         *****************************************************************
         FORMAT   EQU   *
                  ST    R10,SVFORM
                  MVC   OREC,BLANKS
                  MVC   ODESC,IDESC              Description
                  LH    R3,IBEGIN                Beginning inventory
                  CVD   R3,DBL
                  MVC   OBEGIN,WMASK
                  ED    OBEGIN,DBL+6
                  LH    R4,IPURCH                Purchases
                  CVD   R4,DBL
                  MVC   OPURCH,WMASK
                  ED    OPURCH,DBL+6
                  LH    R5,ICALIF                Each product's sales
                  AH    R5,IILL                    by state must be added to
                  AH    R5,IUTAH                     get total for product...
                  AH    R5,IWISC
                  CVD   R5,DBL
                  MVC   OSALES,WMASK
                  ED    OSALES,DBL+6
                  LR    R6,R3                    Ending Inventory =
```

```
                AR    R6,R4                         Beginning + Purchases
                SR    R6,R5                          - Sales
                CVD   R6,DBL
                MVC   OENDING,WMASK
                ED    OENDING,DBL+6
                LH    R3,IQOH                  Actual ending inventory
                CVD   R3,DBL                       (Reusing register 3)
                MVC   OQOH,WMASK                   (Reusing register 3)
                ED    OQOH,DBL+6
                SR    R6,R3                    Difference =
                CVD   R6,DBL                       Expected - Actual
                MVC   ODIFF,WMASK2
                ED    ODIFF,DBL+6
                MVC   OCRLF,WCRLF              PC/370 only.
                L     R10,SVFORM
                BR    R10
*****************************************************************
*       WRITE - Write a single detail line.                     *
*****************************************************************
```

⟨*search..., write REPORT from OREC, no line counting, SVWRITE return* 106⟩

```
          ****************************************************************
          *       WRAPUP - Those things which happen one time only,      *
          *                after all records have been processed.        *
          ****************************************************************
          WRAPUP  EQU   *
                  ST    R10,SVWRAP
                  CLOSE INVENTRY
                  CLOSE REPORT
                  WTO   'COGS14B ... Discrepancies report on REPORT.TXT'
                  L     R10,SVWRAP
                  BR    R10
          ****************************************************************
          *       Literals, if any, will go here                         *
          ****************************************************************
                  LTORG
          ****************************************************************
          *       File definitions                                       *
          ****************************************************************
          INVENTRY DCB  LRECL=28,RECFM=F,MACRF=G,EODAD=ATEND,
                        DDNAME='COGS.BIN'
          REPORT   DCB  LRECL=62,RECFM=F,MACRF=P,
                        DDNAME='REPORT.TXT'
*****************************************************************
*       RETURN ADDRESSES                                        *
*****************************************************************
```

⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩

```
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                   End of file? (Y/N)
BLANKS   DC    CL62' '
WMASK    DC    X'40202120'              BZZ9
WMASK2   DC    X'4020202060'            BZZZ-
DBL      DC    D'0'                     For packed/binary conversions
*****************************************************************
*        Input record definition                                *
*****************************************************************
         DS    0H                       Force halfword alignment
IREC     DS    0CL28          1-28 Inventory record
IDESC    DS    CL10           1-10 Product description
ICALIF   DS    H              11-12 Units sold in Calif
IILL     DS    H              13-14 Units sold in Illinois
IUTAH    DS    H              15-16 Units sold in Utah
IWISC    DS    H              17-18 Units sold in Wisconsin
IBEGIN   DS    H              19-20 Beginning inventory
IPURCH   DS    H              21-22 Purchases throughout year
IQOH     DS    H              23-24 Actual quantity on hand
ICOST    DS    H              25-26 Cost (each) 99V99
ISELL    DS    H              27-28 Sell for (each) 99V99
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62          1-62
ODESC    DS    CL10           1-10 Product description
         DS    CL3            11-13
OBEGIN   DS    CL4            14-17 Beginning inventory
         DS    CL4            18-21
OPURCH   DS    CL4            22-25 Purchases
         DS    CL4            26-29
OSALES   DS    CL4            30-33 Units sold
         DS    CL5            34-38
OENDING DS     CL4            39-42 Ending inventory (expected)
         DS    CL4            43-46
OQOH     DS    CL4            47-50 Ending inventory (actual)
         DS    CL4            51-54
ODIFF    DS    CL5            55-59 Difference
         DS    CL1            60-60
OCRLF    DS    CL2            61-62 PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
```

```
   HD1      DS    0CL62
            DC    CL40'                                    COGSWORTH INDUSTRIES'
            DC    CL20' '
            DC    XL2'0D25'
 HD2     DS    0CL62
         DC    CL40'                Inventory Discrepancies R'
         DC    CL20'eport'
         DC    XL2'0D25'
HD3      DS    0CL62
         DC    CL60' '
         DC    XL2'0D25'
HD4      DS    0CL62
         DC    CL40'Product       Begin + Purch - Sales = Exp'
         DC    CL20'ect    Actual   Diff'
         DC    XL2'0D25'
HD5      DS    0CL62
         DC    CL40'-----    ---   ---   ---   --'
         DC    CL20'--    ---   --'
         DC    XL2'0D25'
         END   BEGIN
```

14.8. Viewing a Register While Testing

We will now see how to view a register while using PC/370's TEST facility. Consider the following, which was presented earlier as an exercise:

Given FIVE DC H'5' , THREE DC H'3' , and RESULT DC X'40202120' , write a short program which will subtract THREE from FIVE giving RESULT . Use WTO to display RESULT . Define other fields as necessary, but do all arithmetic in binary; that is, use packed fields only as required for the ED command.

A solution to this problem is shown here:

```
            START 0
             REGS
BINARY       BEGIN
             LH      R7,FIVE
             SH      R7,THREE
             CVD     R7,DBL
             ED      RESULT,DBL+6
             WTO     RESULT
             RETURN
DBL          DC      D'0'
FIVE         DC      H'5'
THREE        DC      H'3'
RESULT       DC      X'40202120'
```

```
                    END     BINARY
```

Recall that the .PRN listing is required for any testing session. That listing is shown on the next page. We will run a test session, stopping before the LH, SH, CVD, ED, and WTO instructions to view register 7. Within TEST , the r command will display all sixteen registers. We also make use of the l command to limit the trace; that is, by specifying a limit of one, the trace will step through the program one instruction at a time. The test session follows.

```
BINARY                                                          PAGE       1
 PC/370 CROSS ASSEMBLER OPTIONS=LXACE
    LOC                 ADR1     ADR2 LINE       LABEL          OP             OPERANDS
 000000                                    1                     START          0
 000000                                    2   *+++++++          REGS
 000000              00000000              3   R0                EQU            0
 000000              00000001              4   R1                EQU            1
 000000              00000002              5   R2                EQU            2
 000000              00000003              6   R3                EQU            3
 000000              00000004              7   R4                EQU            4
 000000              00000005              8   R5                EQU            5
 000000              00000006              9   R6                EQU            6
 000000              00000007             10   R7                EQU            7
 000000              00000008             11   R8                EQU            8
 000000              00000009             12   R9                EQU            9
 000000              0000000A             13   R10               EQU            10
 000000              0000000B             14   R11               EQU            11
 000000              0000000C             15   R12               EQU            12
 000000              0000000D             16   R13               EQU            13
 000000              0000000E             17   R14               EQU            14
 000000              0000000F             18   R15               EQU            15
 000000                                   19   *++++++           BEGIN
 000000                                   20   BINARY            CSECT
 000000                                   21                     USING          *,15
 000000 47F0F058                 0058     22                     B              KZHQX002
 000004 0B                                23                     DC             AL1(11)
 000005 C2C9D5C1D9E84040                  24                     DC             CL11'BINAR
 000010 0000000000000000                  25   HZQKX002          DC             18F'0'
 000058 90ECD00C                 000C     26   KZHQX002          STM            14,12,12(1
 00005C 50D0F014                 0014     27                     ST             13,HZQKX00
 000060 18ED                              28                     LR             14,13
 000062 41D0F010                 0010     29                     LA             13,HZQKX00
 000066 50D0E008                 0008     30                     ST             13,8(0,14)
 00006A                                   31                     DROP           15
 00006A                                   32                     USING          HZQKX002,1
 00006A 4870D090                 00A0     33                     LH             R7,FIVE
```

```
 00006E 4B70D092                00A2       34                    SH              R7,THREE
 000072 4E70D088                0098       35                    CVD             R7,DBL
 000076 DE03D094D08E   00A4     009E       36                    ED              RESULT,DBL
 00007C                                    37   *++++++++        WTO         RESULT
 00007C 4300D098                00A8       38                    IC              0,RESULT+L
 000080 925BD098                00A8       39                    MVI             RESULT+L'R
 000084 4120D094                00A4       40                    LA              2,RESULT
 000088 0AD1                               41                    SVC             209  00008
 00008E                                    43   *++++++++        RETURN
 00008E 58DD0004                0004       44                    L               13,4(13)
 000092 98ECD00C                000C       45                    LM              14,12,12(1
 000096 07FE                               46                    BR              14
 000098 4000000000000000                   47   DBL              DC              D'0'
 0000A0 0005                               48   FIVE             DC              H'5'
 0000A2 0003                               49   THREE            DC              H'3'
 0000A4 40202120                           50   RESULT           DC              X'40202120
 000000                                    51                    END             BINARY
A:\>binary
T TRACE EP A=07AB ID=370    370 A=000200 OP=47F0F058
  (Copyright message appears here)
TYPE H FOR HELP
+a
ADDR STOP ON
A=26a      00026A 4870D090 4B70D092 4E70D088 DE03D094 .......k+..h...m
T(A-ADDR, E-DATA =, OR N-DATA <>)= a
+t
TRACE SET TRACE EP A=1433 ID=BC     370 A=000200 OP=47F0F058
TRACE EP A=1F9B ID=STM    370 A=000258 OP=90ECD00C
TRACE EP A=17D1 ID=ST     370 A=00025C OP=50D0F014
TRACE EP A=0CAD ID=LR     370 A=000260 OP=18ED
TRACE EP A=1649 ID=LA     370 A=000262 OP=41D0F010
TRACE EP A=17D1 ID=ST     370 A=000266 OP=50D0E008
ADDR STOP      00026A 4870D090 4B70D092 4E70D088 DE03D094 .......k+..h...m
TRACE EP A=1676 ID=LH     370 A=00026A OP=4870D090
+r
 R8-F 00000000 00000000 00000000 00000000 00000000 00000210 00000138 00000200
 PSW 070C00000000026A ILC=4 CC=0
     000266 50D0E008 4870D090 4B70D092 4E70D088 &.\........k+..h
+l
TRACE LIMIT= 1
+t
TRACE SET LIMIT COUNT TRACE EP A=177D ID=SH     370 A=00026E OP=4B70D092
+r
R8-F 00000000 00000000 00000000 00000000 00000000 00000210 00000138 00000200
 PSW 070C00000000026E ILC=4 CC=0
     00026A 4870D090 4B70D092 4E70D088 DE03D094 .......k+..h...m
```

```
+t
TRACE SET LIMIT COUNT
TRACE EP A=14EF ID=CVD    370 A=000272 OP=4E70D088
+r
 R8-F 00000000 00000000 00000000 00000000 00000000 00000210 00000138 00000200
 PSW 070C200000000272 ILC=4 CC=2
     00026E 4B70D092 4E70D088 DE03D094 D08E4300 ...k+..h...m.... +l
TRACE LIMIT= 999
+t
TRACE SET TRACE EP A=2127 ID=ED     370 A=000276 OP=DE03D094D08E
TRACE EP A=161C ID=IC     370 A=00027C OP=4300D098
TRACE EP A=1A85 ID=MVI    370 A=000280 OP=925BD098
TRACE EP A=1649 ID=LA     370 A=000284 OP=4120D094
TRACE EP A=26A3 ID=SVC    370 A=000288 OP=0AD1
   2     (this 2 is from the WTO)
TRACE EP A=1807 ID=STC    370 A=00028A OP=4200D098
TRACE EP A=162D ID=L      370 A=00028E OP=58DD0004
TRACE EP A=19D5 ID=LM     370 A=000292 OP=98ECD00C
TRACE EP A=0B4C ID=BCR    370 A=000296 OP=07FE
TRACE EP A=26A3 ID=SVC    370 A=000102 OP=0A1B
A:\>
```

Summary of Binary Load Instructions

```
Copy the value of the fullword FW to register Rx,
      L  Rx,FW                        maintaining sign integrity.
                        Copy the value of the halfword HW to register Rx,
      LH Rx,HW                        maintaining sign integrity.
                        Copy the value of register Ry to register Rx. Register
      LR Rx,Ry                        Ry remains unchanged.
```

Summary of Binary Compare Instructions

```
Arithmetic compare of the contents of register Rx
      C  Rx,FW                        with the fullword FW
                        Arithmetic compare of the contents of register Rx
      CH Rx,HW                        with the halfword HW
                        Arithmetic compare of the contents of registers Rx
      CR Rx,Ry                        and Ry
```

Summary of Binary Store Instructions

```
Copies the value of register Rx to the fullword FW.
```

```
      ST  Rx,FW
                        Copies the value of register Rx to the halfword HW,
      STH Rx,HW                  truncating if necessary and without warning.
```

Summary of Binary Add Instructions

```
Add the value of the fullword FW to the value in
      A  Rx,FW                   register Rx, with the sum in register Rx.
                        Add the value of the halfword HW to the value in
      AH Rx,HW                   register Rx, with the sum in register Rx.
                        Add the value in register Ry to the value in register
      AR Rx,Ry                   Rx, with the sum in register Rx.
```

Summary of Binary Subtraction Instructions

```
Subtract the value of the fullword FW from the value
      S  Rx,FW
                        in register Rx, with the difference in register Rx.
                        Subtract the value of the halfword HW from the value
      SH Rx,HW
                        in register Rx, with the difference in register Rx.
                        Subtract the value in register Ry from the value in
      SR Rx,Ry
                        register Rx, with the difference in register Rx.
```

14.9. Ejercicios

1. Verdadero o falso. Teniendo en cuenta H1 y H2 son medias palabras, F1 y F2 son fullwords, y D1 y D2 son doublewords ...

   a) Si la F1 comienza en el byte LOC = 02010C continuación, el último de la F1 es en LOC = 02010F. F T

   b) Si H1 comienza en LOC = 020100 y D1 se define inmediatamente después de la H1, a continuación, D1 F T comienza en la LOC = 020108.

   c) Si los campos se definen en el siguiente orden - D1, D2, F1, F2, H1, H2 - continuación F T colectivamente, que ocupan 32 bytes.

   d) Todos los doublewords son fullword alineados, pero no todos son fullwords palabra doble F T alineados.

   e) La secuencia correcta de las instrucciones para mover H1 a la F1 es LH, ST. F T

   f) La secuencia correcta de las instrucciones para mover F1 a H1 es L, ST. F T

g) La secuencia correcta de las instrucciones para añadir H1 a H2 es LH, LH, A, STH. F T

h) La secuencia correcta de las instrucciones para restar H1 de F2 es LH, S, STH. F T

i) Para restar el valor de R3 desde el valor de uso R4 S R4, R3. F T

j) Para mover el valor de R3 a R5 uso ST R5, R3. F T

k) Para mover el número 5 para R6 uso CVB R6 = P'5 ' F T

l) La secuencia correcta de las instrucciones para convertir H1 a un campo de relleno de tamaño PL3 F T es LH, CVD, ZAP.

m) La secuencia correcta de las instrucciones para mover un campo de relleno de PL5 tamaño a la F1 es F T ZAP, CVB, ST.

2. Habida cuenta de la F1, F2 y F3 son fullwords. Escriba el código de BAL a:

   a) lugar de la suma (F1 + F2) en la categoría F3.

   b) lugar de la diferencia (F1-F2) en la categoría F3.

   c) lugar de la suma (F1 +5) en la categoría F3.

   d) lugar el mayor de la F1, F2 y F3 en el registro 4.

   e) el lugar de la suma (F1 + F2 + F3) en el registro 5.

   f) lugar de la suma editado (F1 + F2 + F3) en WK8, utilizando la máscara = X'4020202020202120.

3. Teniendo en cuenta H1, H2, y H3 son medias palabras. Escriba el código de BAL a:

   a) el lugar de la suma (H1 + H2) en el H3.

   b) lugar de la diferencia (H1-H2) en el H3.

   c) el lugar de la suma (H1 +5) en el H3.

   d) lugar la mayor de las H1, H2, y H3 en el registro 4.

   e) el lugar de la suma (H1 + H2 + H3) en el registro 5.

   f) lugar de la suma editado (H1 + H2 + H3) en WK8, utilizando la máscara = X'4020202020202120.

4. Habida cuenta de ZD es un número decimal dividido en zonas (sin embalaje) campo definido como CL3 y HW es una media palabra. Escribir el código de BAL a:

   a) a. lugar de la suma (ZD + HW) en el registro de 4. Además de utilizar binarios. (Usted tendrá que convertir ZD a binario.)

b) b. lugar de la suma (ZD + HW) en PK3, un campo de Bolsas de tres bytes. Uso Bolsas de adición. (Usted tendrá que convertir a HW de pic nic.)

5. (Similar al ejercicio 8 del capítulo 7) columnas 8-14 de una tarjeta-archivo de imagen contiene la saldo de la cuenta del cliente. Mostrar el proceso y las secciones Wrapup de un programa que mostrará el número de clientes y el total (suma) de los saldos. No toda la aritmética en binario, es decir, el uso envasados sólo los campos que sean necesarias para el comando ed. Mostrar todos sobre el terreno definiciones.

6. (Similar al Ejercicio 11 del capítulo 7) Escribir un programa que muestra un recuento de la número de cursos ofrecidos en W93 semestre. Utilice el archivo de OFERTA de la pequeña ciudad Comunidad de base de datos College. La salida debería ser por la OMC sólo: no hay ningún archivo de salida. Utilice un registro en el mostrador. Uso Bolsas de campos sólo como necesaria para el comando de la disfunción eréctil. Su mensaje debe aparecer como sigue: Había cursos ofrecidos en XXX W93 semestre.

7. Consulte COGS9A.MLC en el capítulo 9. Este programa produce la venta Crónica de Dindón Industrias. Vuelva a escribir este programa para que se lea COGS.BIN en lugar de COGS.DAT. ¿Todos los aritméticas en binario, es decir, el uso envasados sólo los campos que sean necesarias para el comando ed.

8. Consulte la base de datos Small Town Hardware Store en más conjuntos de datos. Escriba un programa que va a convertir los campos numéricos en la herramienta de archivo binario. Todos los campos de conteo debe ser almacena como medias palabras. Todos los dólares y los campos ciento deben ser almacenados como fullwords. La salida archivo debe ser dejado en forma EBCDIC. No utilice CR / LF. Llame a su TOOL.BIN archivo.

   a) Nota: puede ser necesario volver a ordenar los campos dentro del registro, o añadir campos no utilizados (relleno), a fin de garantía de que el fullwords y medias palabras pueden, de hecho, estar debidamente alineados.

9. Escribir un programa que lea el archivo producido en el ejercicio 8 anterior y crear el el informe muestra en el ejercicio 14 (a) del Capítulo 7. A pesar de que el programa no se pedía editada salida, use el comando ED para suprimir ceros a la izquierda. ¿Todos los aritméticas en binario, es decir, Bolsas de usar sólo los campos que sean necesarias para el comando de la disfunción eréctil.

10. Ejercicios 10. (Similar al Ejercicio 14 (b) del Capítulo 7) Escribir un programa que lea el archivo producido en el ejercicio 8 anterior y se creará un archivo nueva herramienta con la cantidad sobre la orden en el campo

actualización para aquellos elementos que están ordenados. La nueva cantidad en el orden debe ser igual a la antigua la cantidad de la orden más la cantidad de orden económico. Utilice DDNAME = 'NEWTOOL.DAT' en el DCB para este nuevo archivo (que tiene el mismo diseño de registro, incluyendo fullwords y medias palabras). Todas las herramientas deben ser escritos para este nuevo archivo, incluso aquellos para los que no había orden nuevo colocado.

Capítulo 15

Tabla de Procesamiento

Objetivos Al término de este capítulo usted será capaz de:

- Definir una tabla de datos,
- Código de centinela ciclo controlado,
- Código de un contra-ciclo controlado,
- Usar la LA y BCT instrucciones,
- carga una tabla en tiempo de ejecución,
- Usar técnicas de procesamiento de mesa a un proceso de campo de un carácter por vez, y
- Usar técnicas de procesamiento de mesa para buscar proceso de creación.

Introducción BAL no es probablemente el primer lenguaje de programación que ha aprendido. En su trabajo, usted probablemente han utilizado matrices o tablas. Usted es probablemente consciente de los beneficios del procesamiento de mesa capacidades en un lenguaje de programación. Las aplicaciones más comunes incluyen:

- Al validar (edición) de datos, a menudo queremos asegurarnos de que un campo es una de varias valores admisibles. Estos valores legítimos pueden ser colocados en una tabla, y el campo en cuestión se puede comparar a cada entrada en la tabla. Gastos de envío puede variar según la zona, y cada estado tiene asignado un número de zona.
- El campo de estado dentro de cada registro puede ser utilizado para buscar un estado / zona de la tabla para encontrar la zona correspondiente, y que la zona podría ser utilizado para buscar una zona o un tipo de tabla para encontrar el costo de envío adecuado.

- Un usuario puede solicitar un informe (similares a las ventas Cogsworths 'Crónica) con una columna que muestra por ciento-a-total para cada detalle. Con el fin de elaborar un informe, el total debe ser conocido.

Esto supondrá que o bien pasar el archivo una vez sólo para determinar el total, o el almacenamiento de todos los los datos necesarios en un cuadro que se imprime en EOF cuando se conoce el total. La capacidad de los cuadros proceso requerirá un importante grado de competencia en el trabajo con registros. En este capítulo, vamos a revisar la carga de mesa y la lógica de búsqueda en la tabla, e introducir la LA y las instrucciones de BCT.

15.1. Definición de una tabla; Sentinel ciclos controlados

Considere el siguiente ejemplo: la salida de la mayoría de programas de negocios contendrá registro cuenta para fines de auditoría y la reconciliación. La siguiente salida es de un programa de actualización:

```
Transactions In          74
Transactions Rejected     4
Old Masters In        1,690
Old Masters Deleted      12
Old Masters Changed      21
New Masters Added        37
New Masters Out       1,715
```

¿Cómo podemos utilizar este código de lo que ya sabemos? Una solución sería la de definir la los siguientes campos:

```
#TRANSIN DC PL4'0'   Transactions In
#REJECTS DC PL4'0'   Transactions Rejected
#OLDIN   DC PL4'0'   Old Masters In
#DELETED DC PL4'0'   Old Masters Deleted
#CHANGED DC PL4'0'   Old Masters Changed
#ADDED   DC PL4'0'   New Masters Added
#NEWOUT  DC PL4'0'   New Masters Out
EDCOUNT  DC X'40206B2020206B202120'
OLINE    DS 0CL37
ODESC    DS CL25
OCOUNT   DS CL10    BZ,ZZZ,ZZ9
OCRLF    DC X'0D25'
```

... y escribir lo siguiente (redundante) Código:

```
MVC   ODESC,=CL25'Transactions In'
MVC   OCOUNT,EDCOUNT
```

```
ED    OCOUNT,#TRANSIN'
BAL   R10,WRITE
MVC   ODESC,=CL25'Transactions Rejected'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#REJECTS
BAL   R10,WRITE
MVC   ODESC,=CL25'Old Masters In'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#OLDIN
:
: etc.
:
MVC   ODESC,=CL25'New Masters Out'
MVC   OCOUNT,EDCOUNT
ED    OCOUNT,#NEWOUT
BAL   R10,WRITE
```

Tal código redundante puede estar bien si sólo tiene siete cargos de imprimir como en el ejemplo anterior, pero lo que si usted tiene uno o más cuenta para cada estado?

Lo que se necesita es una manera de repetir el mismo código pero con diferentes cargos y etiquetas, es decir, alguna forma de proceso de estos aspectos como parte de una tabla o matriz. Por ejemplo, en BASIC, lo anterior se podría haber hecho de la siguiente manera. (Este programa utiliza un carácter especial (un asterisco) para indicar el final de la tabla. Cuando un personaje es usado de tal manera que se refiere a veces como un centinela.)

```
100 REM Initialize Counts - In BASIC, numerics
110 REM are automatically initialized to zero.
120 DIM DESC$(8), COUNT(7)
130 DESC$(1) = "Transactions In"
140 DESC$(2) = "Transactions Rejected"
150 DESC$(3) = "Old Masters In"
160 DESC$(4) = "Old Masters Deleted"
170 DESC$(5) = "Old Masters Changed"
180 DESC$(6) = "New Masters Added"
190 DESC$(7) = "New Masters Out"
200 DESC$(8) = "*"
500 REM Print counts 510 MASK$ = "\                          \ ##,###"
520 LET I = 1 520 WHILE DESC$(I) <> "*"
530      PRINT USING MASK$; DESC$(I), COUNT(I)
540      LET I = I + 1
550 WEND
```

¿qué, entonces, es la solución en ensamblador? En primer lugar, queremos definir los campos para que los datos aparece como una mesa. Una solución podría ser:

```
0                    24 25 28
Transactions In         0
Transactions Rejected   0
Old Masters In          0
Old Masters Deleted     0
Old Masters Changed     0
New Masters Added       0
New Masters Out         0
*
```

Cada fila ocupa 29 bytes. El 0, 24, 25 y 28 se refieren a los desplazamientos en una fila. El código para definir como una tabla es como sigue:

```
COUNTS   DS 0CL29
         DC CL25'Transactions In'
#TRANSIN DC PL4'0'
         DC CL25'Transactions Rejected'
#REJECTS DC PL4'0'
         DC CL25'Old Masters In'
#OLDIN   DC PL4'0'
         DC CL25'Old Masters Deleted'
#DELETED DC PL4'0'
         DC CL25'Old Masters Changed'
#CHANGED DC PL4'0'
         DC CL25'New Masters Added'
#ADDED   DC PL4'0'
         DC CL25'New Masters Out'
#NEWOUT  DC PL4'0'
         DC CL1'*'
```

Para incrementar estos cargos, simplemente utilice el comando, como de costumbre, por ejemplo:

```
AP #TRANSIN,=P'1'
```

Para imprimir estos cargos, se utiliza el siguiente código:

```
     LA  R8,COUNTS      R8 POINTS TO FIRST ROW
LOOP EQU *
     MVC ODESC,0(R8)    DESC COMES FROM WHERE R8 POINTS
     MVC OCOUNT,EDCOUNT MOVE MASK PRIOR TO EDIT
     ED  OCOUNT,25(R8)  COUNT COMES FROM 25 BYTES PAST
     BAL R10,PRINT      WHERE R8 POINTS
     AH  R8,=H'29'      POINT TO NEXT ROW
     CLI 0(R8),C'*'     IS FIRST BYTE AN ASTERISK?
     BNE LOOP           NO - REPEAT UNTIL THEN
```

Explicación:

LA...R8,COUNTS Carga la dirección de los cargos en el registro 8, es decir, registro de 8 puntos ?? para el primer byte de la primera entrada en la tabla. Pasar a la descripción de los datos a los que se registran 8 está señalando.

MVC...ODESC,0(R8) El cero indica un desplazamiento de cero, es decir, el registro de 8 puntos directamente al campo. Recordemos que se determina la longitud de un MVC por la longitud del campo de recepción, por lo tanto, no hay longitud el operador en el segundo operando. Editar en el recuento del número de pic nic situado a 25 octetos pasado en el que ??

ED...OCOUNT,25(R8) Registro 8 está señalando. Recordemos que se determina la longitud de una ED por la longitud del campo de recepción, por lo tanto, no hay longitud el operador en el segundo operando.

AH...R8,=H'29' Añadir 29 para colocarse 8, es decir, registrar el punto 8 a la siguiente fila en el ?? mesa. A

CLI...0(R8),C'*' ver si el byte al que registran 8 apunta es un asterisco. Recall ?? que las instrucciones de inmediato (MVI y CLI) no utilizan una longitud de Operador: la duración de estas instrucciones es siempre uno.

intenta

1. ¿Qué cambios se harían a la definición de la tabla y el segmento de programa, si al final de la mesa fue indicado por un signo de dólar en lugar de un asterisco? La palabra final, en lugar de una asterisco? X'FF 'en vez de un asterisco? ¿Qué son estos campos (signo de dólar, END, y X'FF) llama cuando se usa de esta manera?

2. Cada cuenta se define como PL4, lo que significa que cada uno puede tener hasta 9.999.999. ¿Qué cambios Sería necesario que la definición de tabla y el segmento de programa, si cada cuenta se ha cambiado de PL4 a PL3? Indique la nueva gama de valores para estos cargos.

15.2. Contador de ciclos controlados

Se podría muy bien haber aplicado esta como un contador de bucle controlado. En la base, esto habría aparecen como sigue:

```
500 REM Print counts
510 MASK$ = "\                      \ ##,###"
520 FOR I = 1 TO 7
530     PRINT USING MASK$; DESC$(I), COUNT(I)
540 NEXT I
```

La aplicación de BAL de un contador de ciclo controlado se puede hacer de la siguiente manera. Los comentarios en línea debería ser suficiente para explicar el código.

```
     LA  R8,COUNTS       R8 POINTS TO FIRST ROW
     LH  R7,=H'7'        THERE ARE SEVEN ROWS
LOOP EQU *
     MVC ODESC,0(R8)     DESC COMES FROM WHERE R8 POINTS
     MVC OCOUNT,EDCOUNT MOVE MASK PRIOR TO EDIT
     ED  OCOUNT,25(R8)   COUNT COMES FROM 25 BYTES PAST
     BAL R10,PRINT       WHERE R8 POINTS
     AH  R8,=H'29'       POINT TO NEXT ROW
     SH  R7,=H'1'        SUBTRACT ONE FROM ROW COUNTER
     BH  LOOP            IF RESULT > ZERO THEN REPEAT
```

* * * * * * * * * * * * * * * * * * * *

Es útil señalar que podríamos haber codificado el cuadro que cuenta como sigue:

```
COUNTS   DS 0CL29
#TRANSIN DC PL4'0',CL25'Transactions In'
#REJECTS DC PL4'0',CL25'Transactions Rejected'
#OLDIN   DC PL4'0',CL25'Old Masters In'
#DELETED DC PL4'0',CL25'Old Masters Deleted'
#CHANGED DC PL4'0',CL25'Old Masters Changed'
#ADDED   DC PL4'0',CL25'New Masters Added'
#NEWOUT  DC PL4'0',CL25'New Masters Out'
```

- Hemos eliminado el centinela en favor de la lucha contra el control de bucle.
- Cada uno de los campos de conteo (# transindividuales, rechaza #, etc) son consideradas por el ensamblador para ser de cuatro bytes de longitud, ya que es la longitud del campo por primera vez en que PL4 línea (). Esto puede ser verificado por ver las "cosas de la izquierda" para una instrucción hace referencia a estos campos.
- El 0CL29 DS utiliza en cuenta no es obligatorio, pero es una manera conveniente para indicar la duración de la ?? cada fila. Nosotros utilizamos un multiplicador de cero, puesto que los recuentos en sí no ocupan ningún espacio.

Nuestra nueva solución es la siguiente:

```
     LA  R8,COUNTS       R8 POINTS TO FIRST ROW
     LA  R7,7            THERE ARE SEVEN ROWS
LOOP EQU *
     MVC ODESC,4(R8)     DESC FROM 4 PAST WHERE R8 POINTS
```

```
        MVC OCOUNT,EDCOUNT  MOVE MASK PRIOR TO EDIT
        ED  OCOUNT,0(R8)    COUNT COMES FROM WHERE R8 POINTS
        BAL R10,PRINT      LA  R8,L'COUNTS(R8) POINT TO NEXT ROW
        BCT R7,LOOP         SUBTRACT ONE FROM ROW COUNTER
*                           IF RESULT > ZERO THEN REPEAT
```

Explicación:

LA...R7,7 De carga en el registro 7, la dirección del byte séptimo en la memoria. Pero ?? la dirección del byte séptimo es de siete, por lo que el número siete es cargado en el registro 7. Esta es una manera conveniente de poner una constante en un registro.

LA...R8, L'COUNTS(R8) De carga en el registro 8, la dirección del campo que es (longitud de ?? CONDES) o (29) octetos pasado, en registro 8 se señala actualmente. En Dicho de otro modo, apuntan a la siguiente fila. (Esta es la razón por la que utiliza en 0CL29 CUENTA arriba.)

BCT...R7,LOOP El BCT o sucursal en el cómputo de instrucciones es muy común en la tabla ?? transformación. Piense en ello como el equivalente a un BAL DE .. siguiente ciclo. Cada vez que el BCT es ejecutado, se decrementa el registro especificado por uno. Si el resultado es mayor que cero, entonces el programa se ramifica a la etiqueta especificada.

intenta

1. 3. ¿Qué cambios se harían a la definición de la tabla y el segmento de programa, si una cuenta nueva se agregaron a la mesa. El campo nombre es # TRANSOK y la descripción es "Transacciones utilizados. El nuevo campo de seguir el campo # rechaza.

2. 4. Cada cuenta se define como PL4. ¿Qué cambios deben introducirse en la definición de tabla y de segmento del programa, si cada cuenta se ha cambiado de PL4 a PL5?

15.3. Cargando una tabla en tiempo de ejecución

En el ejemplo anterior, el contenido de la tabla (es decir, el recuento y etiquetas) eran conocidos en el momento de El programa fue escrito. No es raro tener que construir una tabla mientras que el programa es funcionamiento. Considere el siguiente escenario. Columnas 1-6 de un disco contiene un título (Sr., Sra.., Etc) Necesito un recuento de cada título que aparece en el archivo, pero no tengo idea de qué títulos se utilizarán. (Hay, después de todo, algunas abreviaturas muy extraño para los títulos militares!) Voy a dejar de lado el espacio para un número arbitrario de título (digamos de hasta 100.) En cada registro se lee, voy a chequear la tabla para ese título. Si lo encuentra, entonces

voy a añadir 1 a la cuenta para ese título. Si el título IsNot encontrado, entonces Lo añadiré al final de la tabla, y Inicia el conteo para que el título 1. Voy a definir la tabla como sigue:

```
#TTLS    DC H'0'
#MAXTTLS DC H'100'
TITLES   DS 100CL10
*           Positions 1-6 are title, CL6
*           Positions 7-10 are count, PL4
```

El código para buscar en la tabla (y añadir, si es necesario) es la siguiente. (Esta es probable que aparezcan en algún lugar dentro de la sección de proceso del programa.)

```
LA     R6,TITLES         Point R6 to start of table
         LH     R7,#TTLS          Nbr table entries to R7
         CH     R7,=H'0'          No entries yet?
         BE     NOTFOUND          Then add title to table
LOOK     EQU    *
         CLC    0(6,R6),ITITLE    Is table title EQ input?
         BE     FOUND             Yes - Increment count
         LA     R6,L'TITLES(R6)   No - Point to next row
         BCT    R7,LOOK           Repeat till all entries
         LH     R7,#TTLS          Reload actual nbr entries
NOTFOUND EQU    *                 Want to add title to table
         CH     R7,#MAXTTLS       Is table already full?
         BNL    TOOMANY           Yes - then message
         MVC    0(6,R6),TITLE     No - then add title
         ZAP    6(4,R6),=P'1'     Initialize count to one
         LA     R7,1(R7)          Increment nbr entries
         STH    R7,#TTLS          and save
         B      LOOKED            Done with new entry
TOOMANY EQU     *                 Table not large enough
         WTO    'Too many titles for table'
         RETURN
FOUND    EQU    *                 Found title in table
         AP     6(4,R6),=P'1'     Increment count for title
LOOKED   EQU    *                 Title has been processed
```

Explicación:

MVC...0(6,R6),ITITLE Mueva el título del registro de entrada a la posición a la que ?? registro de 6 está señalando. Recordemos que se determina la longitud de un MVC por la longitud del primer operando. Debemos especificar una longitud en la primer operando, por lo tanto, el 6 en el interior del paréntesis. Recuerde que el siguiente formato: D

(L, B), donde D es el desplazamiento, L es la de longitud, y B es la (base) de registro.

Recordatorio: especificar la longitud sólo si el formato de la instrucción así lo requiere. En un ejemplo anterior, se utilizó MVC ODESC, 4 (R8) a mover la descripción de la tabla a la salida. Un operando de longitud aquí habría sido adecuado, es decir, MVC ODESC, 4 (25, R8), es válido.

ZAP...6(4,R6), =P'1' Mover un lleno hasta los cuatro bytes situado más allá de seis bytes ?? 6 registro donde está señalando. Al igual que en la nota anterior, 6 es el el desplazamiento, 4 es la longitud, y R6 es el registro. El ZAP comando requiere un operador de longitud para ambos operadores, por lo tanto el 4. Valores por defecto (= P'1 'a una longitud de uno.)

El código para imprimir la tabla es la siguiente. (Esta es probable que aparezcan en algún lugar dentro de la Wrapup sección del programa.)

```
     LA  R6,TITLES         Point R6 to start of table
     LH  R7,#TTLS          Nbr table entries to R7
     CH  R7,=H'0'          No entries yet?
     BE  DONE              Then no titles to print
LOOP EQU *
     BAL R10,CHKLNS        Check for full page
     MVC OTITLE,0(R6)      Move title to output
     MVC OCOUNT,=X'40206B2020206B202120'
     ED  OCOUNT,6(R6)      Edit count to output
     BAL R10,WRITE         Write line
     LA  R6,L'TITLES(R6)   Point to next row
     BCT R7,LOOP           Repeat for all titles
DONE EQU *
```

intenta

1. 5. ¿Qué cambios deben introducirse en la definición de tabla y segmentos de programa para permitir a 200 títulos diferentes?

2. 6. ¿Qué cambios deben introducirse en la definición de tabla y segmentos de programa para permitir la los títulos hasta el ocho bytes de longitud?

3. 7. ¿Qué cambios deben introducirse en la definición de tabla y segmentos de programa si el recuento de fueron cambiados de PL4'0 "(relleno) de H'0" (binario)? Nota: la tabla debe ser halfword alineados. Cada cargo sería automáticamente halfword alineados desde los títulos son aún número de longitud (6, u 8 después de un cambio anterior).

4. 8. ¿Qué cambios deben introducirse en la definición de tabla y segmentos de programa si el número de títulos, # TTLS, se cambió de H'0 a PL2'0?

15.4. Los campos de tablas: El procesamiento de un campo de carácter por carácter

En nuestro siguiente ejemplo, vamos a ver cómo manipular un campo de carácter por carácter. Recordemos que la archivo maestro de la base de datos Small Town Community College muestra el nombre del maestro de nombre (última seguido de una coma y las iniciales) y el grado más alto obtenido. Nos gustaría que estos procesos registros, como para producir una etiqueta de correo para cada profesor, es decir, vamos a utilizar un título de DR. (médico), si el grado más alto obtenido es PHD, de lo contrario vamos a utilizar un título de PROF. (profesor). El la ejecución de este programa aparece como sigue:

```
A\:>teach15a
TEACH15A ... Begin execution
TEACH15A ... Input <BENSON, E.T.    PHD >
TEACH15A ... Output <DR. BENSON           >
TEACH15A ... Input <HINCKLEY, G.B.  MBA >
TEACH15A ... Output <PROF. HINCKLEY       >
TEACH15A ... Input <KIMBALL, S.W.   PHD >
TEACH15A ... Output <DR. KIMBALL          >
TEACH15A ... Input <YOUNG, B.       MBA >
TEACH15A ... Output <PROF. YOUNG          >
TEACH15A ... Input <SMITH, J.       MS  >
TEACH15A ... Output <PROF. SMITH >
TEACH15A ... Normal end of program
```

El diagrama de flujo y el código de la rutina de proceso como de la siguiente manera:

```
PROCESS  EQU *
         ST  R10,SVPROC
         MVC ONAME,=CL25' '
         CLC TTDEG,=CL4'PHD'
         BNE PROFESS
         MVC ONAME(3),=CL3'DR.'
         LA  R3,ONAME+4
         B   PROC2
PROFESS EQU  *
         MVC ONAME(5),=CL5'PROF.'
         LA  R3,ONAME+6
PROC2    EQU *
         LA  R4,TTNAME
         LA  R5,L'TTNAME
PROC3    EQU *
         CLI 0(R4),C','
         BE  PROC4
```

```
          MVC 0(1,R3),0(R4)
          LA  R3,1(R3)
          LA  R4,1(R4)
          BCT R5,PROC3
PROC4     EQU *
          MVC WTOINAME,TTNAME
          MVC WTOIDEG,TTDEG
          WTO WTOMSG1
          MVC WTOONAME,ONAME
          WTO WTOMSG2
          BAL R10,READTCH
PROCESSX  EQU *
          L   R10,SVPROC
          BR  R10
```

El programa completo, TEACH15A.MLC, sigue.

379 ⟨*TEACH15A.MLC* 379⟩≡

```
          PRINT NOGEN
*************************************************************
*        FILENAME: TEACH15A.MLC                             *
*        AUTHOR : Bill Qualls                               *
*        SYSTEM : PC/370 R4.2                               *
*        REMARKS : List of teachers with title, such as     *
*                    'DR. BENSON' or 'PROF. HINCKLEY'.      *
*************************************************************
            START 0
            REGS
BEGIN       BEGIN
            WTO    'TEACH15A ... Begin execution'
            BAL    R10,SETUP
MAIN        EQU    *
            CLI    EOFTEACH,C'Y'
            BE     EOJ
            BAL    R10,PROCESS
            B      MAIN
EOJ         EQU    *
            BAL    R10,WRAPUP
            WTO    'TEACH15A ... Normal end of program'
            RETURN
*************************************************************
*        SETUP - Those things which happen one time only,                *
*                  before any records are processed.                     *
*************************************************************
SETUP       EQU    *
            ST     R10,SVSETUP
            OI     TEACHERS+10,X'08' PC/370 ONLY - Convert all
```

```
*                                          input from ASCII to EBCDIC
         OPEN TEACHERS
         BAL    R10,READTCH
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS EQU     *
         ST     R10,SVPROC
         MVC    ONAME,=CL25' '
         CLC    TTDEG,=CL4'PHD'          Is highest degree PHD?
         BNE    PROFESS                  No, then title is 'PROF.'
         MVC    ONAME(3),=CL3'DR.' else title is 'DR.'
         LA     R3,ONAME+4               Will move name starting at
         B      PROC2                    the fifth byte.
PROFESS EQU     *
         MVC    ONAME(5),=CL5'PROF.'
         LA     R3,ONAME+6               Will move name starting at
PROC2    EQU    *                        the seventh byte.
         LA     R4,TTNAME                Point R4 to 1st byte of input
         LA     R5,L'TTNAME              R5 is max nbr characters
PROC3    EQU    *
         CLI    0(R4),C','               Find comma separating name
         BE     PROC4                    from initials? Yes, done.
         MVC    0(1,R3),0(R4)            Else move this letter.
         LA     R3,1(R3)                 Point to next letter output
         LA     R4,1(R4)                 Point to next letter input
         BCT    R5,PROC3                 Repeat till done
PROC4    EQU    *
         MVC    WTOINAME,TTNAME          Show input name
         MVC    WTOIDEG,TTDEG            Show input degree
         WTO    WTOMSG1
         MVC    WTOONAME,ONAME           Show output name
         WTO    WTOMSG2
         BAL    R10,READTCH
PROCESSX EQU    *
         L      R10,SVPROC
         BR     R10
*****************************************************************
*        READTCH - Read a teacher record.                       *
*****************************************************************
READTCH EQU     *
         ST     R10,SVREADT
         GET    TEACHERS,TREC       Read a single teacher record
         B      READTX
```

```
ATENDTCH EQU   *
         MVI   EOFTEACH,C'Y'
READTX   EQU   *
         L     R10,SVREADT
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE TEACHERS
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
TEACHERS DCB   LRECL=29,RECFM=F,MACRF=G,EODAD=ATENDTCH,
               DDNAME='TEACHER.DAT'
*****************************************************************
*        RETURN ADDRESSES                                        *
*****************************************************************
SVSETUP DC     F'0'                SETUP
SVPROC   DC    F'0'                PROCESS
SVREADT DC     F'0'                READTCH
SVWRAP   DC    F'0'                WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFTEACH DC    CL1'N'              End of teacher file? (Y/N)
ONAME    DC    CL25' '             Name with 'DR.' or 'PROF.'
WTOMSG1 DS     0CL41
         DC    CL21'TEACH15A ... Input <'
WTOINAME DS    CL15
WTOIDEG DS     CL4
         DC    CL1'>'
WTOMSG2 DS     0CL47
         DC    CL21'TEACH15A ... Output <'
WTOONAME DS    CL25
         DC    CL1'>'
*****************************************************************
*        Input record definition - Teacher                      *
```

```
*****************************************************************
TREC     DS    0CL29       1-29    Teacher record
TTID     DS    CL3         1- 3    Teacher ID nbr
TTNAME   DS    CL15        4-18    Teacher name
TTDEG    DS    CL4        19-22    Highest degree
TTTEN    DS    CL1        23-23    Tenured?
TTPHONE DS     CL4        24-27    Phone nbr
TTCRLF   DS    CL2        28-29    PC/370 only - CR/LF
         END   BEGIN
```

15.5. Muestra del Programa - la producción de transcripciones

Vamos a incluir otros dos programas en este capítulo. Nuestro objetivo es producir las transcripciones de todos los los estudiantes de la pequeña ciudad universitaria de la Comunidad. Con el fin de hacerlo, necesitamos los siguientes los campos de los siguientes archivos:

```
ESTUDIANTES DE GRADOS DEL CURSO                    ........ .....          ID de
```

Las transcripciones serán producidos en los De secuencia de identificación, para que el estudiante y los archivos GRADO tendrá que ser ordenados en consecuencia. Anexar especie clave para grabar en Además, para cada estudiante, los cursos permiten ordenar por identificación del estudiante, semestre , carta del año semestre (descendente será incluido en la secuencia en la que Ordenar Anexar número así W93 viene después de F92, pero fueron tomadas.

Este es un problema.

El semestre antes de F93), y el ID del curso. consta de una carta y un año, por ejemplo: F92. Pero la caída 1992 se sitúa después del invierno de 1992, por lo que necesitamos el semestre para ser ordenados en la parte del año ascendente, luego de la temporada porción descendente.

Esto no es un problema con un paquete de tipo decente, pero no podemos hacer con tipo de comando de DOS. Así que tipo / clase 14 / 1 clase / 1 necesidad de escribir un programa que añade un especie clave para el archivo de grado. El semestre se a la codificación: W92 será 921 y F92 se 922. También incluirá el curso de identificación en esta clave de ordenación para que dentro de cada semestre, Los cursos están enumerados en secuencia. Para cada uno supuesto que quieren que la descripción del curso. Producir También será necesario las horas del curso en las transcripciones de orden para determinar la hora y el grado de tentativa el promedio de puntos. Por lo tanto, vamos a ordenar la Transcripciones CURSO archivo de ID del curso y la carga en una en el disco como Informe.txt mesa en el programa de transcripción.

Aquí está una lista de los comandos para producir las transcripciones, así como las transcripciones de sí mismos.

```
A:\MIN>transcr1
TRANSCR1 ... Begin execution
TRANSCR1 ... 22 grade records reformatted.
TRANSCR1 ... Normal end of program
A:\MIN>sort /+14 < grade.$$$ > grade.srt
A:\MIN>sort /+1 < student.dat > student.srt
A:\MIN>sort /+1 < course.dat > course.srt
A:\MIN>transcr2
TRANSCR2 ... Begin execution
TRANSCR2 ... Transcripts on REPORT.TXT
TRANSCR2 ... Normal end of program
A:\MIN>type report.txt
------------------------------------------------------------
         TRANSCRIPT FOR (125) MORALES, L.A.
Semester   Course    Description    Grade Hours Points
--------   ------  ---------------  ----- ----- ------
  F92      MA101   ALGEBRA            F        3     0
  W93      MA101   ALGEBRA            C        3     6
--------   ------  ---------------  ----- ----- ------
                                     TOTAL      6     6
                                     GPA     1.00
------------------------------------------------------------
         TRANSCRIPT FOR (263) HAVLIK, K.M.
Semester   Course    Description    Grade Hours Points
--------   ------  ---------------  ----- ----- ------
  W92      PE151   AEROBICS           C        1     2
  F92      PE151   AEROBICS           B        1     3
  W93      PE151   AEROBICS           A        1     4
--------   ------  ---------------  ----- ----- ------
                                     TOTAL      3     9
                                     GPA     3.00
------------------------------------------------------------
         TRANSCRIPT FOR (402) FOOTE, A.K.
Semester   Course    Description    Grade Hours Points
--------   ------  ---------------  ----- ----- ------
                                     TOTAL      0     0
                                     GPA     0.00
------------------------------------------------------------
         TRANSCRIPT FOR (421) QUALLS, G.E.
Semester   Course    Description    Grade Hours Points
--------   ------  ---------------  ----- ----- ------
  F92      EG101   ENGLISH I          B        3     9
  W93      EG102   ENGLISH II         A        3    12
--------   ------  ---------------  ----- ----- ------
                                     TOTAL      6    21
                                     GPA     3.50
```

```
------------------------------------------------------------
         TRANSCRIPT FOR (626) MERCIER, J.L.
Semester   Course    Description    Grade   Hours Points
--------   ------  --------------  -----   ----- ------
  W92      EG102   ENGLISH II        A        3    12
--------   ------  --------------  -----   ----- ------
                                   TOTAL      3    12
                                   GPA     4.00
         TRANSCRIPT FOR (701) ARIAS, I.L.
Semester   Course    Description    Grade Hours Points
--------   ------  --------------  ----- ----- ------
  F92      MA101   ALGEBRA           B       3     9
  F92      PE151   AEROBICS          A       1     4
  W93      MA107   STATISTICS        D       3     3
--------   ------  --------------  ----- ----- ------
                                   TOTAL     7    16
                                   GPA    2.29
------------------------------------------------------------
         TRANSCRIPT FOR (713) HILMER, D.R.
Semester   Course    Description    Grade Hours Points
--------   ------  --------------  ----- ----- ------
  F92      EG101   ENGLISH I         C       3     6
  F92      MA101   ALGEBRA           B       3     9
  W93      EG102   ENGLISH II        B       3     9
  W93      MA107   STATISTICS        B       3     9
--------   ------  --------------  ----- ----- ------
                                   TOTAL    12    33
                                   GPA    2.75
------------------------------------------------------------
         TRANSCRIPT FOR (896) QUALLS, D.M.
Semester   Course    Description    Grade  Hours Points
--------   ------  --------------  -----  ----- ------
  W92      PE151   AEROBICS          A       1     4
  F92      AC101   ACCOUNTING        C       3     6
  F92      BU101   BUSINESS          C       3     6
  F92      EG101   ENGLISH I         A       3    12
  F92      MA101   ALGEBRA           B       3     9
  W93      EG102   ENGLISH II        B       3     9
  W93      MA107   STATISTICS        A       3    12
--------   ------  --------------  -----  ----- ------
                                   TOTAL    19    58
                                   GPA    3.05
```

Lo que sigue es una lista de programa TRANSCR1.MLC, el programa para añadir la clave de ordenación para el archivo de grado. No hay nada nuevo aquí. No hay necesidad de explicación: se incluye de forma complementaria.

385 ⟨ff 40⟩+≡ ◁289 395▷

```
   PRINT NOGEN
*****************************************************************
*        FILENAME: TRANSCR1.MLC                                 *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Reformat grade file so it can be sorted      *
*                    properly so as to produce transcripts.     *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO   'TRANSCR1 ... Begin execution'
         BAL   R10,SETUP
MAIN     EQU    *
         CLI    EOFGRADE,C'Y'
         BE     EOJ
         BAL    R10,PROCESS
         B      MAIN
EOJ      EQU    *
         BAL    R10,WRAPUP
         WTO    'TRANSCR1 ... Normal end of program'
RETURN   EQU    *
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                  before any records are processed.            *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OI     GRADEIN+10,X'08'          PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     GRADEOUT+10,X'08' PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN GRADEIN
         OPEN GRADEOUT
         BAL    R10,READGRAD              Priming read - GRADEIN
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS EQU     *
         ST     R10,SVPROC
         BAL    R10,FORMAT
         BAL    R10,WRITE
```

```
         BAL    R10,READGRAD
PROCESSX EQU    *
         L      R10,SVPROC
         BR     R10
*************************************************************
*        FORMAT - Format a single record, with sort key.    *
*************************************************************
*        Copy the entire record, with sort key appended.                *
*        Sort key consists of student ID, year portion of               *
*        semester, '1' for (W)inter or '2' for (F)all, and              *
*        the course ID. This appended sort key is used to               *
*        overcome limitations of DOS' SORT command, which               *
*        allows a single sort field only. The recoding of               *
*        the semester was done so Fall classes will come                *
*        after winter classes. For example, semester 'W92'              *
*        becomes '921', while semester 'F92' becomes '922'.             *
*************************************************************
FORMAT   EQU    *
         ST     R10,SVFORMAT
         MVC    SORTDATA,GREC
         MVC    SORTSID,GSID
         MVC    SORTSEM(2),GSEM+1
         CLI    GSEM,C'F'
         BNE    FORMAT2
         MVI    SORTSEM+2,C'2'
         B      FORMAT3
FORMAT2  EQU    *
         MVI    SORTSEM+2,C'1'
FORMAT3 EQU     *
         MVC    SORTCID,GCID
         MVC    SORTCRLF,WCRLF
FORMATX EQU     *
         L      R10,SVFORMAT
         BR     R10
*************************************************************
*        READGRAD - Read a Grade record.                    *
*************************************************************
READGRAD EQU    *
         ST     R10,SVREADG
         GET    GRADEIN,GREC
         B      READGX
ATENDGRA EQU    *
         MVI    EOFGRADE,C'Y'
READGX   EQU    *
         L      R10,SVREADG
         BR     R10
```

```
*****************************************************************
*        WRITE - Write a single output record.                  *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   GRADEOUT,SORTREC
         AP    #OUT,=P'1'
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,                *
*                 after all records have been processed.                  *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE GRADEIN
         CLOSE GRADEOUT
         ED    MSG#OUT,#OUT
         WTO   MSG
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                                   *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                                 *
*        Note: $$$ is common DOS extension for temporary file  *
*****************************************************************
GRADEIN DCB    LRECL=15,RECFM=F,MACRF=G,EODAD=ATENDGRA,
               DDNAME='GRADE.DAT'
GRADEOUT DCB   LRECL=26,RECFM=F,MACRF=P,
               DDNAME='GRADE.$$$'
*****************************************************************
*        RETURN ADDRESSES                                                 *
*****************************************************************
SVSETUP DC     F'0'                    SETUP
SVPROC   DC    F'0'                    PROCESS
SVREADG DC     F'0'                    READCOUR
SVWRITE DC     F'0'              WRITE
SVWRAP   DC    F'0'              WRAPUP
SVFORMAT DC    F'0'              FORMAT
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'           PC/370 ONLY - EBCDIC CR/LF
```

```
EOFGRADE DC    CL1'N'             End of grades file? (Y/N)
#OUT     DC    PL2'0'
MSG      DS    0CL43                DC    CL12'TRANSCR1 ...'
MSG#OUT DC     XL4'40202120'
         DC    CL27' grade records reformatted.'
*****************************************************************
*        Input record definition - Grade                        *
*****************************************************************
GREC     DS    0CL15      1-15    Grade record
GSID     DS    CL3        1- 3    Student ID nbr
GSEM     DS    CL3        4- 6    Semester
GCID     DS    CL5        7-11    Course ID nbr
GSECT    DS    CL1       12-12    Section number
GGRADE   DS    CL1       13-13    Grade earned
GGCRLF   DS    CL2       14-15    PC/370 only - CR/LF
*****************************************************************
*        Output record definition - Grade w/ sort key           *
*****************************************************************
SORTREC DS     0CL26      1-26    Sort record
SORTDATA DS    CL13       1-13    Grade record (without CRLF)
SORTKEY DS     0CL11     14-24    Sort key, including:
SORTSID DS     CL3       14-16    Student ID nbr
SORTSEM DS     CL3       17-19    Semester (recoded)
SORTCID DS     CL5       20-24    Course ID nbr
SORTCRLF DS    CL2       25-26    PC/370 only - CR/LF
         END   BEGIN
```

```
Output from TRANSCR1.MLC:
GRADE.$$$ Input to TRANSCR1.MLC: GRADE.DAT          Input to TRANSCR2.MLC
        626W92EG1021A                   626W92EG1021A626921EG102
        896W92PE1511A                   896W92PE1511A896921PE151
        263W92PE1511C                   263W92PE1511C263921PE151
        896F92AC1011C                   896F92AC1011C896922AC101
        896F92BU1011C                   896F92BU1011C896922BU101
        896F92EG1011A                   896F92EG1011A896922EG101
        713F92EG1012C                   713F92EG1012C713922EG101
        421F92EG1012B                   421F92EG1012B421922EG101
        713F92MA1011B                   713F92MA1011B713922MA101
        896F92MA1011B                   896F92MA1011B896922MA101
        125F92MA1012F                   125F92MA1012F125922MA101
        701F92MA1012B                   701F92MA1012B701922MA101
        263F92PE1511B                   263F92PE1511B263922PE151
        701F92PE1511A                   701F92PE1511A701922PE151
        713W93EG1021B                   713W93EG1021B713931EG102
        421W93EG1021A                   421W93EG1021A421931EG102
        896W93EG1021B                   896W93EG1021B896931EG102
        125W93MA1011C                   125W93MA1011C125931MA101
        713W93MA1071B                   713W93MA1071B713931MA107
        896W93MA1071A                   896W93MA1071A896931MA107
        701W93MA1071D                   701W93MA1071D701931MA107
        263W93PE1511A                   263W93PE1511A263931PE151
```

Los siguientes son los diagramas de flujo y el código para TRANSCR2.MLC. Este programa produce las transcripciones. Hay, por diseño, diferentes ejemplos de procesamiento de mesa en el programa.

15.5.0.1. Línea principal de la sección

La sección de la línea principal es el mismo que hemos visto tantas veces antes de ...

```
        START  0
        REGS
BEGIN   BEGIN
        WTO    'TRANSCR2 ... Begin execution'
```

389 ⟨*search..., main loop SETUP,EOFSTUD to EOJ,PROCESS,WRAPUP* 389⟩≡ (395)

```
        BAL    R10,SETUP
MAIN    EQU    *
        CLI EOFSTUD,C'Y'
        BE  EOJ
        BAL    R10,PROCESS
        B      MAIN
```

```
EOJ      EQU    *
         BAL    R10,WRAPUP

          WTO    'TRANSCR2 ... Normal end of program'
RETURN   EQU    *
         RETURN
```

15.6. La rutina de instalación

La rutina de instalación es muy similar a lo que hemos visto antes de. Incluye abre y lee de cebado. Lo nuevo es la lógica para cargar los datos del curso el archivo del campo en un tabla interna. Esa lógica es el objeto de la siguiente rutina ...

```
SETUP EQU  *
      ST   R10,SVSETUP
      BAL  R10,LOADTBL
      OI   STUDENT+10,X'08'
      OI   GRADE+10,X'08'
      OI   REPORT+10,X'08'
      OPEN STUDENT
      OPEN GRADE
      OPEN REPORT
      BAL  R10,READSTUD
      BAL  R10,READGRAD
      L    R10,SVSETUP
      BR   R10
```

15.6.1. Carga de la tabla de golf

Antes de que se procesan los registros de estudiantes, cargamos todos los CURSO registros en una tabla interna. Verificamos que los cursos son en número de secuencia de curso y que no hay espacio suficiente para almacenar todos los cursos

```
LOADTBL  EQU   *
         ST    R10,SVLOAD
         OI    COURSE+10,X'08'
         OPEN  COURSE
         LA    R3,TABLE
         SR    R4,R4
LOADTBL2 EQU   *
         BAL   R10,READCOUR
         CLI   EOFCOURS,C'Y'
         BE    LOADTBLX
```

```
          CLC   PREVCID,CCID
          BNL   LOADTBL3
          MVC   PREVCID,CCID
          CH    R4,MAXROWS
          BNL   LOADTBL4
          MVC   0(5,R3),CCID
          MVC   5(15,R3),CCDESC
          PACK  20(1,R3),CCHRS
          LA    R3,L'TABLE(R3)
          LA    R4,1(R4)
          B     LOADTBL2
LOADTBL3  EQU   *
          WTO   'TRANSCR2 ... Course file not sequenced by CID'
          B     RETURN
LOADTBL4  EQU   *
          WTO   'TRANSCR2 ... Nbr of courses exceeds table  size'
          B     RETURN
LOADTBLX  EQU   *
          STH   R4,ROWS
          CLOSE COURSE
          L     R10,SVLOAD
          BR    R10
```

15.6.2. Las rutinas de LEA

Hay tres archivos de entrada: cursos, los estudiantes, y los grados. Las rutinas para los tres son muy similares ...

```
READCOUR EQU *
         ST  R10,SVREADC
         GET COURSE,CREC
         B   READCX
ATENDCRS EQU *
         MVI EOFCOURS,C'Y'
READCX   EQU *
         L   R10,SVREADC
         BR  R10
```

.

```
READSTUD EQU *
         ST  R10,SVREADS
         GET STUDENT,SREC
         B   READSX
ATENDSTU EQU *
         MVI EOFSTUD,C'Y'
READSX   EQU *
```

```
          L   R10,SVREADS
          BR  R10
.
READGRAD EQU *
          ST  R10,SVREADG
          GET GRADE,GREC
          B   READGX
ATENDGRA EQU *
          MVI EOFGRADE,C'Y'
READGX   EQU *
          L   R10,SVREADG
          BR  R10
```

15.6.3. La rutina PROCESO

Al entrar a la rutina de proceso, tenemos un registro de estudiante. Esta rutina procesará todos los registros de Grado para ese estudiante. Empezamos por producir la transcripción de la partida. Para cada registro de grado, hemos formato de una línea de discurso único y acumular horas de intento y la calidad de los puntos (véase la rutina de formato inferior). Después de procesar todos los registros de Grado para este estudiante, se calcula e imprime el PAM. A continuación, leer el registro siguiente estudiante antes de regresar a la línea principal.

```
FORMAT  EQU *
        ST  R10,SVFORMAT
        MVC OSEM,GSEM
        MVC OCID,GCID
        MVC OGRADE,GGRADE
        BAL R10,CRSDATA
        MVC OCDESC,5(R3)
        ZAP PK2,20(1,R3)
        MVC OCHRS,=X'40202120'
        ED  OCHRS,PK2
        AP  ATTEMPT,PK2
        BAL R10,GRADEVAL
        ZAP PK4,PK2
        MP  PK4,VALUE
        MVC OPOINTS,=X'40202120'
        ED  OPOINTS,PK4+2
        AP  QUALITY,PK4
        MVC OCRLF,WCRLF
        BAL R10,WRITE
FORMATX EQU *
        L   R10,SVFORMAT
```

```
         BR  R10
```

.

```
WRITE EQU *
      ST  R10,SVWRITE
      PUT REPORT,OREC
      L   R10,SVWRITE
      BR  R10
```

15.6.4. GET Información del Curso de rutina

Esta rutina usa una búsqueda en la tabla para determinar la descripción del curso y las horas de un curso determinado de identificación. Nota: esta búsqueda utiliza un contador de bucle controlado.

```
CRSDATA  EQU *
         ST  R10,SVCRSDAT
         LA  R3,TABLE
         LH  R4,ROWS
CRSDATA2 EQU *
         CLC GCID,0(R3)
         BE  CRSDATAX
         LA  R3,L'TABLE(R3)
         BCT R4,CRSDATA2
         WTO 'TRANSCR2 ... Bad course ID in Grade file'
         B   RETURN
CRSDATAX EQU *
         L   R10,SVCRSDAT
         BR  R10
```

donde...

```
TABLE    DS    10CL21
*   1- 5    Course ID
*   6-20    Course description
*  21-21    Hours (packed)
```

15.6.5. OBTENER GRADO DE VALOR rutina

Esta rutina usa una búsqueda en la tabla a la determinar los puntos de calidad para un determinado grado. Nota: esta búsqueda utiliza una bucle controlado centinela.

```
GRADEVAL EQU *
         ST  R10,SVGRDVAL
         LA  R7,GRADETBL
```

```
GRADEVA2 EQU *
         CLC 0(1,R7),GGRADE
         BE  GRADEVAX
         LA  R7,L'GRADETBL(R7)
         CLI 0(R7),X'FF'
         BNE GRADEVA2
         WTO 'TRANSCR2 ... Invalid grade in grade file'
         B   RETURN
GRADEVAX EQU *
         ZAP VALUE,1(1,R7)
         L   R10,SVGRDVAL
         BR  R10
```

donde...

```
VALUE    DC PL1'0'
GRADETBL DS 0CL2
         DC CL1'A',PL1'4'
         DC CL1'B',PL1'3'
         DC CL1'C',PL1'2'
         DC CL1'D',PL1'1'
         DC CL1'F',PL1'0'
         DC X'FF'
```

15.6.6. DO GPA de rutina

Puntos de la calidad Esta rutina se mueve el total de horas intentado y total para este estudiante a la transcripción, determina el Programa de Acción Mundial por los puntos divisorios de la calidad total por el total de horas intentado, y se mueve del PAM a la transcripción.

```
DOGPA      EQU     *
           ST      R10,SVDOGPA
           CP      ATTEMPT,=P'0'
           BE      DOGPA2
           PUT     REPORT,HD4
DOGPA2     EQU     *
           MVC     TATTEMPT,=X'40202120'
           ED      TATTEMPT,ATTEMPT
           MVC     TQUALITY,=X'40202120'
           ED      TQUALITY,QUALITY
           PUT     REPORT,TREC
           ZAP     DIVIDEND,QUALITY
           SRP     DIVIDEND,3,0
           ZAP     DIVISOR,ATTEMPT
           BZ      DOGPA3
           DP      DIVIDEND,DIVISOR
```

```
            SRP     QUOTIENT,64-1,5
            ZAP     PK3,QUOTIENT
            B       DOGPA4
DOGPA3      EQU     *
            ZAP     PK3,=P'0'
DOGPA4      EQU     *
            MVC     WK7,=X'402021204B2020'
            ED      WK7,PK3
            MVC     AGPA,WK7+2
            PUT     REPORT,AREC
DOGPAX      EQU     *
            L       R10,SVDOGPA
            BR      R10
```

15.7. La rutina Wrapup

Cuando las transcripciones de todos los estudiantes se han impreso, se cierran todos los archivos abiertos.

```
WRAPUP EQU   *
       ST    R10,SVWRAP
       CLOSE STUDENT
       CLOSE GRADE
       CLOSE REPORT
       WTO   'TRANSCR2 ... Transcripts on REPORT.TXT'
       L     R10,SVWRAP
       BR    R10
```

Para completar, el código fuente completo para el programa anotado TRANSCR2.MLC siguiente:

395 ⟨*ff* 40⟩+≡ ◁385 436▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: TRANSCR2.MLC                                 *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce transcipts.                          *
*                  This program illustrates table logic.        *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'TRANSCR2 ... Begin execution'
⟨search..., main loop SETUP,EOFSTUD to EOJ,PROCESS,WRAPUP 389⟩
         WTO    'TRANSCR2 ... Normal end of program'
```

```
RETURN   EQU    *
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,                *
*                 before any records are processed.                      *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         BAL    R10,LOADTBL
         OI     STUDENT+10,X'08'          PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     GRADE+10,X'08'            PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN STUDENT
         OPEN GRADE
         OPEN REPORT
         BAL    R10,READSTUD              Priming read - STUDENT
         BAL    R10,READGRAD              Priming read - GRADE
         L      R10,SVSETUP
         BR     R10
*****************************************************************
*        LOADTBL - Load all courses into table. Verify:                  *
*            (1) courses are in course nbr sequence, and                 *
*            (2) there is enough room in table for all.                  *
*****************************************************************
LOADTBL EQU     *
         ST     R10,SVLOAD
         OI     COURSE+10,X'08'           PC/370 ONLY - Convert all
*                                         input from ASCII to EBCDIC
         OPEN COURSE
         LA     R3,TABLE                  Point to start of table
         SR     R4,R4                     Initialize row count to zero
LOADTBL2 EQU    *
         BAL    R10,READCOUR              Read single course record
         CLI   EOFCOURS,C'Y'             At end?
         BE    LOADTBLX                  Yes - Load complete.
         CLC   PREVCID,CCID              Sequence check
         BNL   LOADTBL3                  Fatal error...
         MVC   PREVCID,CCID              Save course ID for seq check
         CH    R4,MAXROWS                Table full already?
         BNL   LOADTBL4                  Yes - Fatal error...
         MVC   0(5,R3),CCID              Move course ID nbr,
         MVC   5(15,R3),CCDESC             course description, and
         PACK 20(1,R3),CCHRS                 course hours to row.
```

```
         LA    R3,L'TABLE(R3)            Point to next row
         LA    R4,1(R4)                  Increment row count
         B     LOADTBL2                  Repeat
LOADTBL3 EQU   *
         WTO   'TRANSCR2 ... Course file not sequenced by CID'
         B     RETURN
LOADTBL4 EQU   *
         WTO   'TRANSCR2 ... Nbr of courses exceeds table size'
         B     RETURN
LOADTBLX EQU   *
         STH   R4,ROWS
         CLOSE COURSE
         L     R10,SVLOAD
         BR    R10
*****************************************************************
*        HDGS - Print headings.                                 *
*****************************************************************
HDGS     EQU   *
         ST    R10,SVHDGS
         MVC   HDSID,SSID                Move student ID to first hdg
         MVC   HDSNAME,SSNAME            Move student name to first hdg
         PUT   REPORT,FORMFEED           PC/370 ONLY
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.  *
*****************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         BAL   R10,HDGS                  Start student on a new page
         ZAP   ATTEMPT,=P'0'             Init hrs attempted to zero
         ZAP   QUALITY,=P'0'             Init quality pts to zero
PROC2    EQU   *                         Check for student ID found
         CLI   EOFGRADE,C'Y'             on GRADE but not on STUDENT.
         BE    PROC3                     This is a serious error,
         CLC   GSID,SSID                 but for this program we will
         BNL   PROC3                     just skip all such records.
         BAL   R10,READGRAD
         B     PROC2
PROC3    EQU   *                         Process all grades records
         CLI   EOFGRADE,C'Y'             for the current student.
         BE    PROC4
```

```
         CLC   GSID,SSID
         BNE   PROC4
         BAL   R10,FORMAT
         BAL   R10,READGRAD
         B     PROC3
PROC4    EQU   *                       Having processed any and all
         BAL   R10,DOGPA               grades records for the current
         BAL   R10,READSTUD            student, do GPA line and read
PROCESSX EQU   *                       the next student.
         L     R10,SVPROC
         BR    R10
*************************************************************
*        FORMAT - Format a single transcript line.                    *
*************************************************************
FORMAT   EQU   *
         ST    R10,SVFORMAT
         MVC   OSEM,GSEM               Move semester,
         MVC   OCID,GCID                 course ID nbr, and
         MVC   OGRADE,GGRADE               grade earned to output
         BAL   R10,CRSDATA             Find course data in table
         MVC   OCDESC,5(R3)            Course desc comes from table
         ZAP   PK2,20(1,R3)
         MVC   OCHRS,=X'40202120'
         ED    OCHRS,PK2               Course hours comes from table
         AP    ATTEMPT,PK2             Accumulate hours attempted
         BAL   R10,GRADEVAL
         ZAP   PK4,PK2
         MP    PK4,VALUE
         MVC   OPOINTS,=X'40202120'
         ED    OPOINTS,PK4+2
         AP    QUALITY,PK4
         MVC   OCRLF,WCRLF
         BAL   R10,WRITE
FORMATX  EQU   *
         L     R10,SVFORMAT
         BR    R10
*************************************************************
*        CRSDATA - Find course data in table                          *
*************************************************************
CRSDATA  EQU   *
         ST    R10,SVCRSDAT
         LA    R3,TABLE
         LH    R4,ROWS
CRSDATA2 EQU   *
         CLC   GCID,0(R3)
         BE    CRSDATAX
```

```
         LA    R3,L'TABLE(R3)
         BCT   R4,CRSDATA2
         WTO   'TRANSCR2 ... Bad course ID in Grade file'
         B     RETURN
CRSDATAX EQU   *
         L     R10,SVCRSDAT
         BR    R10
*****************************************************************
*        GRADEVAL - Find point value for grade                  *
*****************************************************************
GRADEVAL EQU   *
         ST    R10,SVGRDVAL
         LA    R7,GRADETBL                 Point to start of table
GRADEVA2 EQU   *
         CLC   0(1,R7),GGRADE              Compare grade to the grade
         BE    GRADEVAX                    in table. If equal, done.
         LA    R7,L'GRADETBL(R7)           Else point to next row
         CLI   0(R7),X'FF'                 See if at end of table.
         BNE   GRADEVA2                    No, repeat.
         WTO   'TRANSCR2 ... Invalid grade in grade file'
         B     RETURN                      Fatal error...
GRADEVAX EQU   *
         ZAP   VALUE,1(1,R7)               Save grade value
         L     R10,SVGRDVAL
         BR    R10
*****************************************************************
*        DOGPA - Calculate and format GPA                       *
*****************************************************************
DOGPA    EQU   *
         ST    R10,SVDOGPA
         CP    ATTEMPT,=P'0'
         BE    DOGPA2
         PUT   REPORT,HD4
DOGPA2   EQU   *
         MVC   TATTEMPT,=X'40202120'
         ED    TATTEMPT,ATTEMPT
         MVC   TQUALITY,=X'40202120'
         ED    TQUALITY,QUALITY
         PUT   REPORT,TREC
         ZAP   DIVIDEND,QUALITY
         SRP   DIVIDEND,3,0
         ZAP   DIVISOR,ATTEMPT
         BZ    DOGPA3
         DP    DIVIDEND,DIVISOR
         SRP   QUOTIENT,64-1,5
         ZAP   PK3,QUOTIENT
```

```
         B     DOGPA4
DOGPA3   EQU   *
         ZAP   PK3,=P'0'
DOGPA4   EQU   *
         MVC   WK7,=X'402021204B2020'
         ED    WK7,PK3
         MVC   AGPA,WK7+2
         PUT   REPORT,AREC
DOGPAX   EQU   *
         L     R10,SVDOGPA
         BR    R10
*****************************************************************
*        READSTU - Read a student record.                       *
*****************************************************************
READSTUD EQU   *
         ST    R10,SVREADS
         GET   STUDENT,SREC
         B     READSX
ATENDSTU EQU   *
         MVI   EOFSTUD,C'Y'
READSX   EQU   *
         L     R10,SVREADS
         BR    R10
*****************************************************************
*        READCRS - Read a course record.                        *
*****************************************************************
READCOUR EQU   *
         ST    R10,SVREADC
         GET   COURSE,CREC
         B     READCX
ATENDCRS EQU   *
         MVI   EOFCOURS,C'Y'
READCX   EQU   *
         L     R10,SVREADC
         BR    R10
*****************************************************************
*        READGRAD - Read a Grade record.                        *
*****************************************************************
READGRAD EQU   *
         ST    R10,SVREADG
         GET   GRADE,GREC
         B     READGX
ATENDGRA EQU   *
         MVI   EOFGRADE,C'Y'
READGX   EQU   *
         L     R10,SVREADG
```

```
         BR    R10
*****************************************************************
*        WRITE - Write a single detail line.                    *
*****************************************************************
WRITE    EQU   *
         ST    R10,SVWRITE
         PUT   REPORT,OREC              Write report line
         L     R10,SVWRITE
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         CLOSE STUDENT
         CLOSE GRADE
         CLOSE REPORT
         WTO   'TRANSCR2 ... Transcripts on REPORT.TXT'
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
STUDENT DCB    LRECL=22,RECFM=F,MACRF=G,EODAD=ATENDSTU,
               DDNAME='STUDENT.SRT'
COURSE   DCB   LRECL=23,RECFM=F,MACRF=G,EODAD=ATENDCRS,
               DDNAME='COURSE.SRT'
GRADE    DCB   LRECL=26,RECFM=F,MACRF=G,EODAD=ATENDGRA,
               DDNAME='GRADE.SRT'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP DC     F'0'                    SETUP
SVHDGS   DC    F'0'                    HDGS
SVPROC   DC    F'0'                    PROCESS
SVREADS DC     F'0'                    READSTUD
SVREADC DC     F'0'                    READGRAD
SVREADG DC     F'0'                    READCOUR
SVLOAD   DC    F'0'                    LOADTBL
```

```
SVWRITE DC     F'0'                        WRITE
SVWRAP   DC    F'0'                        WRAPUP
SVFORMAT DC    F'0'                        FORMAT
SVCRSDAT DC    F'0'                        CRSDATA
SVGRDVAL DC    F'0'                        GRADEVAL
SVDOGPA DC     F'0'                        DOGPA
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                     PC/370 ONLY - EBCDIC CR/LF
EOFSTUD DC     CL1'N'                      End of students file? (Y/N)
EOFCOURS DC    CL1'N'                      End of course file? (Y/N)
EOFGRADE DC    CL1'N'                      End of grades file? (Y/N)
POINTS   DC    PL2'0'                      Points for this course
QUALITY DC     PL2'0'                      Total points
ATTEMPT DC     PL2'0'                      Hours attempted
ACCUM    DC    PL2'0'                      Accumulated points
PREVCID DC     XL5'00'                     Sequence check on course ID
WK7      DC    CL7' '
PK2      DC    PL2'0'
PK3      DC    PL3'0'
PK4      DC    PL4'0'
         COPY DIVISION
*****************************************************************
*        Table to determine value of a letter grade.            *
*****************************************************************
VALUE    DC    PL1'0'                      How much this grade is worth
GRADETBL DS    0CL2
         DC    CL1'A',PL1'4'
         DC    CL1'B',PL1'3'
         DC    CL1'C',PL1'2'
         DC    CL1'D',PL1'1'
         DC    CL1'F',PL1'0'
         DC    X'FF'
*****************************************************************
*        Input record definition - Student                      *
*****************************************************************
SREC     DS    0CL22         1-22          Student record
SSID     DS    CL3           1- 3          Student ID nbr
SSNAME   DS    CL15          4-18          Student name
SSSEX    DS    CL1          19-19          Gender
SSMAR    DS    CL1          20-20          Marital status
SSCRLF   DS    CL2          21-22          PC/370 only - CR/LF
*****************************************************************
*        Input record definition - Course                       *
*****************************************************************
```

```
CREC     DS    0CL23         1-23         Course record
CCID     DS    CL5           1- 5         Course ID nbr
CCDESC   DS    CL15          5-20         Course description
CCHRS    DS    CL1          21-21         Hours
CCCRLF   DS    CL2          22-23         PC/370 only - CR/LF
*****************************************************************
*        Input record definition - Grade                        *
*****************************************************************
GREC     DS    0CL26         1-26         Grade record
GSID     DS    CL3           1- 3         Student ID nbr
GSEM     DS    CL3           4- 6         Semester
GCID     DS    CL5           7-11         Course ID nbr
GSECT    DS    CL1          12-12         Section number
GGRADE   DS    CL1          13-13         Grade earned
GKEY     DS    CL11         14-24         Sort key (see TRANS1.MLC)
GGCRLF   DS    CL2          25-26         PC/370 only - CR/LF
*****************************************************************
*        Course table                                           *
*****************************************************************
ROWS     DC    H'0'                       Entries in course table
MAXROWS DC     H'10'                      Max entries in course table
TABLE    DS    10CL21                     Each row consists of:
*                            1- 5         Course ID
*                            6-20         Course description
*                           21-21         Hours (packed)
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62         1-62         Report record
         DC    CL2' '        1- 2
OSEM     DS    CL3           3- 5         Semester
         DC    CL5' '        6-10
OCID     DS    CL5          11-15         Course ID
         DC    CL3' '       16-18
OCDESC   DS    CL15         19-33         Course Description
         DC    CL4' '       34-37
OGRADE   DS    CL1          38-38         Grade
         DC    CL4' '       39-42
OCHRS    DS    CL4          43-46         Course Hours (BZZ9)
         DC    CL3' '       47-49
OPOINTS DS     CL4          50-53         Quality Points (BZZ9)
         DC    CL7' '       54-60
OCRLF    DS    CL2          61-62         PC/370 only - CR/LF
*****************************************************************
*        Totals line definition                                 *
*****************************************************************
```

```
TREC     DS    0CL62         1-62
         DC    CL35' '       1-35
         DC    CL7'TOTAL'
TATTEMPT DS    CL4          43-46         Course Hours (BZZ9)
         DC    CL3' '       47-49
TQUALITY DS    CL4          50-53         Quality Points (BZZ9)
         DC    CL7' '       54-60
         DC    X'0D25'      61-62         PC/370 only - CR/LF
*****************************************************************
*        Totals line definition                                 *
*****************************************************************
AREC     DS    0CL62         1-62
         DC    CL35' '       1-35
         DC    CL6'GPA'
AGPA     DS    CL5          42-46         GPA (B9.99)
         DC    CL14' '      54-60
         DC    X'0D25'      61-62         PC/370 only - CR/LF
*****************************************************************
*        Headings definitions                                   *
*****************************************************************
FORMFEED DS    0CL62                      PC/370 only
*        DC    X'0C'                      EBCDIC formfeed
*        DC    CL59' '
         DC    60C'_'                     For testing...
         DC    X'0D25'                    EBCDIC CR/LF
HD1      DS    0CL62
         DC    CL24'            TRANSCRIPT FOR ('
HDSID    DS    CL3                   DC    CL2') '
HDSNAME DS     CL15
        DC     CL16' '
        DC     XL2'0D25'
HD2     DS     0CL62
        DC     CL60' '
        DC     XL2'0D25'
HD3     DS     0CL62
        DC     CL40'Semester Course   Description   Grade'
        DC     CL20' Hours Points     '
        DC     XL2'0D25'
HD4     DS     0CL62
        DC     CL40'---- --- --------
        DC     CL20' --- ---     '
        DC     XL2'0D25'
        END    BEGIN
```

15.7.1. La automatización del proceso con un archivo por lotes

Esto no pretende ser un libro en el PC / MS-DOS. Sin embargo, el lector que es competente en PC reconocerá que un archivo por lotes puede (y debe) ser usado para simplificar la ejecución de este programa. Mi solución siguiente:

```
A:\MIN>type transcr.bat
@echo off rem TRANSCR.BAT by BQ - Produce transcripts
rem Append sort key to grade record...
transcr1
echo Sorting the reformatted grade file...
sort /+14 < grade.$$$ > grade.srt
rem Temporary file is no longer needed...
del grade.$$$
echo Sorting the student file...
sort /+1 < student.dat > student.srt
echo Sorting the course file...
sort /+1 < course.dat > course.srt
rem Produce transcripts... transcr2
echo Cleaning up...
del grade.srt
del student.srt
del course.srt
echo Done!
A:\MIN>transcr
 TRANSCR1 ... Begin execution
TRANSCR1 ... 22 grade records reformatted.
TRANSCR1 ... Normal end of program
Sorting the reformatted grade file...
Sorting the student file...
Sorting the course file...
TRANSCR2 ... Begin execution
TRANSCR2 ... Transcripts on REPORT.TXT
TRANSCR2 ... Normal end of program
Cleaning up...
Done!
A:\MIN>
```

15.8. Un resumen de las más comunes Cuadro Instrucciones de procesamiento

A punto de un registro a una tabla:

```
LA      R8,TABLE
L       R8,=A(TABLE)
```

Para hacer un recuento en un registro:

```
L       R7,=F'5'
LH      R7,=H'5'
LA      R7,5
```

Para apuntar a la fila siguiente:

```
A       R7,=F'29'
AH      R7,=H'29'
LA      R7,29(R7)
```

Para "copia de seguridad" de un registro, como en el punto a una fila anterior:

```
S       R7,=F'29'
SH      R7,=H'29'
```

Si usted es "una copia de seguridad" por un valor de un (sólo) también se puede utilizar:

```
BCTR   R7,0
```

Para borrar (cero) de un registro:

```
L       R5,=F'0'
LH      R5,=H'0'
SR      R5,R5
SLR     R5,R5
```

15.9. Ejercicios

1. 1. Verdadero o falso. Todas las preguntas se refieren a la serie de sesiones siguiente programa:

```
REGIONS DS 0CL10
EAST    DC PL2'0',CL8'EAST'
MIDWEST DC PL2'0',CL8'MIDWEST'
WEST    DC PL2'0',CL8'WEST'
        DC X'FF'
TOTAL   DC PL3'0'
```

   a) a. El X'FF "al final de la tabla se conoce como un centenario. F T

   b) b. Para añadir 1 al Medio Oeste, podemos usar AP MIDWEST, = P'1 ' F T

   c) c. Para registrar el punto 3 del inicio de la mesa, podemos usar LA R3, REGIONES F T

d) d. Si el registro 3 se apunta a una fila, se puede utilizar LA R3, 10 para que apunte a la siguiente fila. F T

e) e. Si el registro 3 se apunta a una fila, podemos utilizar una R3, = F'10 'para apuntar a la próxima F T fila.

f) f. Si el registro 3 se apunta a una fila, podemos utilizar LA R3, REGIONES L'(R3) para apuntar a F T la siguiente fila.

g) g. Si el registro 3 se apunta a una fila, podemos utilizar SH R3, = H'10 'para apuntar a la F T fila anterior.

h) h. Si el registro 3 se apunta a oeste, a continuación, AP WEST, = P'1 'es el mismo que F T AP 0 (R3), = P'1 '

i) i. Si el registro 3 se apunta a una fila, para añadir que el número de filas de TOTAL, se puede utilizar F T AP TOTAL, 0 (2, R3)

j) j. Para ver si el registro 3 se apunta a la final de la tabla, podemos usar la F T CLI 0 (1, R3), X'FF '

k) k. Si el registro 3 se apunta a una fila, podemos utilizar CP 0 (2, R3), = P'0 'para ver si se F T recuento de filas es igual a cero.

l) l. Si se sustituye a todos PL2'0 con H'0 ', entonces tenemos que asegurarnos es halfword REGIONES F T alineados.

m) m. Si se sustituye a todos PL2'0 con H'0 ', y el registro 3 se apunta a una fila, entonces F T puede utilizar AH 0 (2, R3), = H'1 'para añadir de 1 a contar esa fila.

2. Habida cuenta de las definiciones de los campos siguientes:

```
WSTATE  DS   CL2
FOUNDSW DS   CL1
NO      EQU  C'N'
YES     EQU  C'Y'
#STATES DC   H'50'
STATES  DC   CL
        DC
        etc.
```

Llena los espacios en blanco:

```
* IS WSTATE IN TABLE?
         ____  R8,_______ Point to start of table
         ____  R7,_______ Load nbr of table entries
LOOP     EQU   *          ____  __________ Match?
         ____  __________ Yes - Go to found
         ____  __________ No - point to next entry
         ____  __________ Repeat if not at end
         ____  __________ Indicate not found
         ____  __________ Exit
```

```
FOUND    EQU   *          Match was found
         ____  __________ So indicate
EXIT     EQU   *
```

3. Habida cuenta de las definiciones de los campos siguientes:

```
STREC    DS  0CL7
STSTATE  DS  CL2
STRATE   DS  CL3        (V999)
STCRLF   DS  CL2        PC/370 only
EOFSW    DC  CL1'N'
YES      EQU C'Y'
#ENTRIES DS  H
TAXTABLE DS  50CL4 *            1-2 STATE (Character)
*            3-4 SALES TAX RATE V999 (Packed)
```

Llena los espacios en blanco:

```
* LOAD TABLE W/ STATE AND SALES TAX RATE
         ____ R6,_______ Point to start of table
         ____ R5,_______ Set count to zero
         BAL   R10,READST Get State/Rate record
LOOP     EQU   *
         ____ __________ End of file?
         ____ __________ Yes - Done.
         ____ __________ Move state to table
         ____ __________ Move rate to table
         ____ __________ Point to next entry
         ____ __________ Increment
         ____ __________ Get another record
         ____ __________ Repeat
DONE     EQU   *
         ____ __________ Save counter
```

4. Habida cuenta de las definiciones de los campos siguientes:

```
      DS 0F             Force fullword alignment
IREC  DS 0CL40   1 - 40 Input Record, EBCDIC, no CR/LF
INAME DS CL24    1 - 24 First and last name
ITLY  DS F      25 - 28 Total sales last year Jan-June
IMTY  DS 6H     29 - 40 Monthly sales this year, Jan-June
*
OLINE DS 0CL100  1 -100 Output Line
ONAME DS CL24    1 - 24 First and last name
      DS CL4    25 - 28
OMTY  DS 6CL8   29 - 76 Monthly sales this year, BZZ,ZZ9-
OTTY  DS CL11   77 - 87 Total this year, BZ,ZZZ,ZZ9-
```

```
OTLY  DS CL11   88 - 98 Total last year, BZ,ZZZ,ZZ9-
OCRLF DS CL2    99 -100 PC/370 only - CR/LF
*
DBL   DS D
FULL  DS F
WTTY  DS F
PK3   DS PL3
```

Llena los espacios en blanco:

```
* Determine total sales this year for this employee
         _____ R6,_____          Point to first month
         _____ R5,_____          How many months?
         ____ R4,_____           Use R4 to hold total
LOOP1    EQU   *
         ____ _________________ Add monthly sales to total
         ____ _________________ Point to next month
         ____ _________________ Repeat till all months done
         ____ _____,WTTY         Save Result
* Move Monthly Sales (only) to Output
         ____  R7,_____          Point to first month-Output
         ____  R6,_____          Point to first month-Input
         ____  R5,_____          How many months?
LOOP2    EQU   *
         ____  R3,_____          Place months sales in register
         ____  _________________ Convert to decimal
         ____  PK3,_____________ Move to smaller field
         ____  _________________                        Move mask
         ____  _________________ Edit amount
         ____  _________________ Point to next month-Output
         ____  _________________ Point to next month-Input
         ____  _________________ Repeat till all months done
```

5. (Consulte la base de datos Small Town Banco de Sangre en más conjuntos de datos.) Anote el nombre de los donantes en la tabla de los donantes es almacenado con el nombre y apellido es separar los campos. Escribir una breve programa (similar a TEACH15A.MLC en este capítulo) que muestra cada nombre de los donantes como un solo campo, es decir, con un único espacio que separa el nombre y apellido.

6. (Consulte el auto de pueblo pequeño de almacenamiento en la base de datos más conjuntos de datos.) Nota del cliente nombre en la tabla CUST se almacena como apellido, seguido de una coma, seguido por la primera nombre, por ejemplo: Cecilia Quilty,. Escribir un programa corto (similar a TEACH15A.MLC en de este capítulo) que muestra el nombre de cada cliente con el nombre del primero es lo primero y no coma; por ejemplo: Cecilia Quilty.

7. (Consulte la base de datos Small Town Banco de Sangre en más conjuntos de datos.) Crear una detallada la historia de las donaciones. Para cada uno de los donantes, la lista de la fecha de cada donación y el nombre del grupo de acredita con la donación. Su informe debe aparecer como sigue:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345
                     SMALL TOWN BLOOD BANK              Page BZZ9
                     Detailed Donor History
ID#      Donor Name          Type        Group            Date
--- -------------------- ---- --------------------   --------
XXX XXXXXXXXXX XXXXXXXXXX XXX XXXXXXXXXXXXXXXXXXX    mm/dd/yy
                                  XXXXXXXXXXXXXXXXXXX mm/dd/yy
XXX XXXXXXXXXX XXXXXXXXXX XXX    XXXXXXXXXXXXXXXXXXX mm/dd/yy
                                  XXXXXXXXXXXXXXXXXXX mm/dd/yy
```

8. (Consulte la base de datos Small Town Hardware Store en más conjuntos de datos.) Elaborar una lista de kits y sus herramientas como sigue:

```
.        1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345
                     SMALL TOWN HARDWARE STORE          Page BZZ9
                            Kits List
          KIT                          TOOL
-------------------------   -------------------------
ID#      Description         ID#      Description         Cost
--- ----------------------   --- ---------------------- --------
XXX XXXXXXXXXXXXXXXXXXXX   XXX XXXXXXXXXXXXXXXXXXXX BZZZ.99
                           XXX XXXXXXXXXXXXXXXXXXXX BZZ9.99
                               TOTAL COST             BZZ9.99
XXX XXXXXXXXXXXXXXXXXXXX   XXX XXXXXXXXXXXXXXXXXXXX BZZZ.99
                           XXX XXXXXXXXXXXXXXXXXXXX BZZ9.99
                               TOTAL COST             BZZ9.99
```

Tendrá que usar el kit, maquillaje, y los archivos de herramientas. Cargue el archivo de herramienta en una tabla. Coincidir y de maquillaje en kit de identificación. Utilice la herramienta de identificación en el archivo de maquillaje para buscar en la tabla para encontrar el KIT coincidía con la descripción y el costo.

Capítulo 16

Más aritmética binaria

Objetivos Al término de este capítulo usted será capaz de:

- Definir incluso pares impares de los registros,
- Usar la M, MH, MR y las instrucciones para realizar la multiplicación binaria, y la
- Usar la D y las instrucciones para llevar a cabo DR división binaria.

Introducción En el capítulo 14 vimos cómo sumar y restar números binarios. En concreto, nos fijamos en la A, AH, AR, S, SH, y las instrucciones de SR. Recuerde que para cada una de estas instrucciones, el primer operando es siempre un registro, y el segundo operando sea una fullword, media palabra, o regístrese. En este capítulo vamos a completar nuestro estudio de las matemáticas binarias mirando a la multiplicación y la las instrucciones de la división: M, MH, MR, D, y DR.

16.1. Par-impar Registro de Pares

Con el fin de discutir binarios, o registro, multiplicación, debemos introducir un nuevo concepto - el de incluso pares impares de los registros. Recuerde que hay registros de dieciséis en total, numerados del 0 al 15. Esto nos da ocho pares de pares impares de registros:

```
R0 R1 R2 R3 R4 R5 R6 R7
R8 R9 R10 R11 R12 R15 R14 R13
```

Recuerde, incluso un par impar de registros se inicia con el registro de número par, que es el más bajo de los dos números. Por ejemplo, (R2, R3) es un par extraña pareja, pero (R3, R4) y (R5, R6) se ni siquiera los pares impares.

16.2. Binary Multiplicación: La M, MH, MR, y las instrucciones

La M (multiplicar) la instrucción se multiplica un registro por un fullword. Por ejemplo:

```
               where SUBTOTAL
L R9,SUBTOTAL
                DS F
               where DISCOUNT
M R8,DISCOUNT
                DS F
```

- En primer lugar, uno de los valores que se multiplican se coloca en el registro de número impar de un par-impar ?? par.
- En segundo lugar, se multiplican. Especificar el registro de números pares como el primer operando, y el otro ?? fullword como el segundo operando. (Si el primer operando no es un registro de pares, es obtendrá una excepción de la especificación en tiempo de ejecución.)
- Por último, el producto será una palabra doble, incluso ocupando la par impar. (Por lo general el derecho ?? la mayoría de parte del producto, es decir, el registro de número impar, será suficiente para sostener el producto. Recuerde que un registro único puede contener un valor de hasta 2.147.483.647. Si es necesario, Usted puede utilizar el almacén de la instrucción múltiple para almacenar productos muy importantes en una palabra doble, para ejemplo, STM, R4, R5, DBL. Asimismo, puede utilizar la instrucción de carga múltiple para poner un palabra doble en un par de registros, por ejemplo, LM R4, R5, DBL.

No tenemos que estar preocupados con lo que está en el número par de registro antes de que se multiplican. Ello no necesita ser inicializada todo lo que es su lugar será sustituido por los dígitos de orden superior de la producto.

Veamos más de cerca otro ejemplo. Teniendo en cuenta:

```
FULL1 DC F'64' 00 00 00 40
FULL2 DC F'8'  00 00 00 08
```

Para multiplicar FULL1 por FULL2 (64 x 8 = 512 = X'200 ') y convertir el producto a un número de Bolsas de que el código (asumir DBLWORD DC D'0 '):

```
R4                R5 L R5,FULL1 ?? ??   ?? ??      00 00    00 40
                 R4                R5 M R4,FULL2 00 00   00 00      00 00    02 00
                     DBLWORD
```

```
                        00  00 00
                                                                    R4 R5
          L R5, FULL1?? ?? ?? ?? 00 00 00 40
                                                                     R4 R5
          M R4, FULL2 00 00 00 00 00 00 02 00
```

También podemos utilizar el MR (multiplicar registro) la instrucción para multiplicar incluso un par de extraños por un (único) registro. Por ejemplo:

```
                   R4            R5
L  R5,FULL1 ?? ??   ?? ?? 00 00    00 40
L  R6,FULL2                    R4            R5
MR R4,R6    00 00   00 00 00 00    02 00
```

intenta

1. 1. Teniendo en cuenta: X DC F'16 ', Y DC F'3', Y y Z DC F'0 '. Suministro de las instrucciones para multiplicar x por y, con el producto en Z y en el registro 7. Mostrar los resultados intermedios.

```
R6 R7                                                                R6 R7
```

1. 2. Teniendo en cuenta: Un F'32 DC, DC F'4 B, y C DC F'0. Suministro de las instrucciones para multiplicar A por B, con el producto en C y en el registro 5. Mostrar los resultados intermedios.

```
R4 R5                                                                R4 R5
```

1. 3. Teniendo en cuenta: registro de 9 contiene 64. Utilice la instrucción M para multiplicar esto por 5, con el producto en el Registro 9. Mostrar los resultados intermedios.

```
R8 R9          (antes)
```

1. Teniendo en cuenta: registro 5 contiene 4, registro de 6 contiene 3, y registro de 7 contiene 2. Suministro de la la instrucción se multiplica el valor en el registro de 7 por el valor en el registro de 5, con el producto en el registro 7. Mostrar los resultados intermedios.

```
R6 R7          (antes)
```

* * * * * * * * * * * * * * * * * * * *

En estos dos ejemplos, el segundo operando era pequeño (8 y 3). Cada uno de ellos, de hecho, cabe en un media palabra. Si uno o ambos de los operandos

de una instrucción de multiplicación es una media palabra, el MH (multiplicar) halfword instrucción puede ser utilizado. La instrucción de MH es mucho más simple que el M o MR: se utiliza un registro de un solo, y este registro puede ser par o impar. Por ejemplo:

```
                where SUBTOTAL
L  R8,SUBTOTAL
                DS F
                where DISCOUNT
MH R8,DISCOUNT
                DS H
```

- En primer lugar, uno de los valores que se multiplican se coloca en un registro. Recordatorio: el uso L si el valor es un ?? fullword, o LH, si el valor es una media palabra.
- En segundo lugar, se multiplican. Especificar el registro como el primer operando, y la media palabra como el segundo ?? operando.
- Por último, el producto será un fullword ocupando el registro. (Si el producto no caben en el ?? registro, el truncamiento se produce sin previo aviso.)

Considere el siguiente ejemplo:

```
FULL1 DC F'64' 00 00 00 40
HALF2 DC H'8'  00 08
```

Para multiplicar FULL1 por HALF2 (64 x 8 = 512 = X'200 ') y convertir el producto a un número de Bolsas de que el código:

```
                    R4 L   R4,FULL1   00 00   00 40
                    R4 MH  R4,HALF2   00 00   02 00
                         DBLWORD
CVD R4,DBLWORD 00 00
  00 00  00 00 51 2C
```

intenta

1. 5. Teniendo en cuenta: R H'15 DC, DC H'4 S, y T DC H'0. Suministro de las instrucciones para multiplicar por S R dando T. Mostrar los resultados intermedios.

```
R3                                                              R3
```

1. 6. Teniendo en cuenta: Un F'32 DC, DC H'4 B, y C DC H'0. Suministro de las instrucciones para multiplicar A por B da C. Mostrar los resultados intermedios.

R4 R4

1. 7. Teniendo en cuenta: registro de 6 contiene 16. Utilice la instrucción MH a multiplicar este por 4, con el producto en el registro de 6. Mostrar los resultados intermedios.

R5 R6 (antes) R5

16.3. Binary División: El D y DR Instrucciones

Recuerde que en nuestro análisis de la DP (brecha de relleno) la instrucción, tras la división de la dividendo fue sustituido por el cociente (a la izquierda) y el resto (a la derecha). Algo similar ocurre con la división de registro. Al igual que con la multiplicación registro, la división de registro utiliza un incluso-extraña pareja de registros. Inicialmente, el dividendo se ocupan incluso un par de extraños. A raíz de la operación de división, el cociente será en un registro, y el resto será en la otra registro. Hay un punto potencial de confusión: a diferencia de la brecha Bolsas de instrucción, la el cociente será a la derecha (en el registro impar) y el resto estará en la izquierda (incluso en el registro).

El D (dividir) la instrucción se divide el dividendo (incluso en un par-impar) por un fullword. Por ejemplo:

```
           where
L R9,SUM
        SUM   DS F
M R8,=F'1'
           where
D R8,COUNT
      COUNT DS F
```

- En primer lugar, el dividendo debe ser colocado en el registro de número impar de un par de par-impar. ?? Recordatorio: el uso L si el dividendo es un fullword, o LH si el dividendo es una media palabra.
- En segundo lugar, a diferencia de la multiplicación binaria, el contenido del registro de pares es importante ?? para la división. Usualmente se desea eliminar (cero). Lo hacemos multiplicando la extraña pareja-incluso por un fullword con un valor de uno. (Recuerde que para multiplicar que incluso especificar el registro de la incluso par-impar, y el producto va a ocupar el par.)
- En tercer lugar, se dividen. Especificar el registro de números pares como el primer operando y un fullword ?? que contiene el divisor es el segundo operando. (Si el primer operando no es un número par de registro, recibirá una excepción de la especificación en tiempo de ejecución.)

- Por último, el cociente será en el registro de número impar y el resto será en el aún ?? registro numerado.

Como se mencionó anteriormente, el contenido del registro de pares es importante, es decir, tanto registros en el par determinar el valor del dividendo. Reconociendo esto, no es raro que ver a alguien a cero el registro de números pares, restando de sí misma, como en SR R8, R8, pero esto no funcionará si el dividendo es negativo. En su lugar, multiplicar el par por un fullword de uno (como se muestra arriba) con el fin de mantener la integridad de la señal.

Echemos un vistazo más de cerca el ejemplo de arriba. Teniendo en cuenta:

```
SUM   DC F'214' 00 00 00 D6
COUNT DC F'8'   00 00 00 08
AVG   DC PL3'0' 00 00 0C
```

Para dividir SUM por el Conde dar PROMEDIO (214 / 8 = 26 + resto 6)

```
                          R8                      R9
L   R9,SUM       ??    ??   ??   ??       00   00    00 D6
                          R8                      R9
M   R8,=F'1'     00    00   00   00       00   00    00 D6
                          R8                      R9
D   R8,COUNT     00    00   00   06       00   00    00 1A
                     remainder                quotient
                                DBLWORD
CVD R9,DBLWORD   00     00   00   00   00 00   02    6C
                       AVG
ZAP AVG,DBLWORD  00     02   6C
```

¿Qué pasa si queremos que el resultado se redondea? Es decir, 214 / 8 = 26,75, que nos gustaría mostrar como 27. Con el fin de hacerlo, en lugar de multiplicar el dividendo por uno, vamos a multiplicar por diez. Nosotros entonces tiene 2140 / 8 = 267, que podemos cambiar-y todo el año para obtener 27. Por ejemplo:

```
                             R8                      R9
L   R9,SUM           ??    ??   ??   ??       00   00    00 D6
M   R8,=F'10'        00    00   00   00       00   00    08 5C
                             R8                      R9
D   R8,COUNT         00    00   00   04       00   00    01 0B
                        remainder                quotient
                                   DBLWORD
CVD R9,DBLWORD       00     00   00   00   00 00   26    7C
                                   DBLWORD
SRP DBLWORD,64-1,5   00     00   00   00   00 00   02    7C
                          AVG
ZAP AVG,DBLWORD      00     02   7C
```

También puede utilizar el DR (registro de dividir) la instrucción para dividir un par extraña pareja por un (único) registro:

```
                        R8              R9
L  R9,SUM     ??    ??   ?? ?? 00   00    00 D6
                        R8              R9
M  R8,=F'10' 00     00   00 00 00   00    08 5C
L  R7,COUNT
                        R8              R9
DR R8,R7      00    00   00 04 00   00    01 0B
                  remainder        quotient
```

Nota: no hay equivalente a dividir MH, es decir, no hay media palabra dividir. Esto no quiere decir halfword no se puede utilizar como divisor, simplemente que se debe cargar a un primer registro (con LH), y luego se utiliza la instrucción DR.

intenta

1. 8. Teniendo en cuenta: X DC F'17 ', Y DC H'3', Y y Z DC PL3'0 '. Suministro de las instrucciones para dividir x por y dando Z igual a X'00567C '(que representa 5,67).

```
R4 R5                                                    R4 R5          M
```

1. 9. Teniendo en cuenta: Un F'20 DC, DC F'42 B, y C DC PL3'0. Suministro de las instrucciones para dividir A por B da C igual a X'00048C '(que representa el 48 %). Sugerencia: 20.000 / 42 = 476 + R8.

```
R6 R7                                                    R6 R7          M
```

Para ilustrar estos conceptos se presentan dos programas: COGS16A.MLC y COGS16B.MLC, que se son modificaciones de COGS13A.MLC y COGS13B.MLC respectivamente. Estos programas fueron presentados en el capítulo 13. COGS16A.MLC determinará las ventas en dólares a nivel nacional para las Industrias Dindón, mientras que COGS16B.MLC elaborará un informe que muestra la contribución de California a las ventas. Ambos programas se leer COGS.BIN, que es el equivalente binario a COGS.DAT. COGS.BIN fue creado por COGS14A.MLC como se muestra en el capítulo 14. El listado de programas de seguimiento. Los cambios en las versiones anteriores han sido sombreado. La ejecución y la salida no se muestran como son los mismos que se muestran en el capítulo 13.

16.4. Programa de ejemplo: las ventas en dólares a nivel nacional Dindón's

418 ⟨*COGS16A.MLC* 418⟩≡

```
                PRINT NOGEN
***************************************************************
*        FILENAME: COGS16A.MLC                                *
*        AUTHOR : Bill Qualls                                 *
*        SYSTEM : PC/370 R4.2                                 *
*        REMARKS : Determine nationwide dollar sales for      *
*                     COGSWORTH INDUSTRIES.                   *
*                     This is a modification of COGS13A.MLC and*
*                     illustrates binary multiplication.      *
***************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'COGS16A ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'COGS16A ... Normal end of program'
         RETURN
***************************************************************
*        SETUP - Those things which happen one time only,     *
*                 before any records are processed.           *
***************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OPEN INVENTRY                      Input is EBCDIC, no CR/LF
         BAL    R10,READ
         L      R10,SVSETUP
         BR     R10
*************************************************************
*        READ - Read a record.                              *
*************************************************************
⟨search..., read INVENTRY into IREC, EOFSW, SVREAD return 314⟩
***************************************************************
*        PROCESS - Those things which happen once per record. *
***************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         LH    R3,ICALIF                 Determine total units
         AH    R3,IILL                     sold for this product
         AH    R3,IUTAH
         AH    R3,IWISC
         MH    R3,ISELL                  Multiply units by price
```

```
         A     R3,TOTAL                  Add total thus far
         ST    R3,TOTAL                    then save back.
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        WRAPUP - Those things which happen one time only,      *
*                 after all records have been processed.        *
*****************************************************************
WRAPUP   EQU   *
         ST    R10,SVWRAP
         L     R3,TOTAL                  Must put it in a register
         CVD   R3,DBLWORD                  to convert it to packed.
         ED    ODOLLARS,DBLWORD+4
         WTO   OMSG
         CLOSE INVENTRY
         L     R10,SVWRAP
         BR    R10
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=28,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.BIN'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
SVSETUP DC     F'0'                      SETUP
SVPROC   DC    F'0'                      PROCESS
SVREAD   DC    F'0'                      READ
SVWRAP   DC    F'0'                      WRAPUP
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
EOFSW    DC    CL1'N'                    End of file? (Y/N)
TOTAL    DC    F'0'                      Nationwide dollar sales
DBLWORD DC     D'0'
*****************************************************************
*        Input record definition                                *
*****************************************************************
         DS    0H                        Force halfword alignment
IREC     DS    0CL28           1-28 Inventory record
IDESC    DS    CL10            1-10 Product description
```

```
   ICALIF   DS    H           11-12 Units sold in Calif
IILL     DS    H           13-14 Units sold in Illinois
IUTAH    DS    H           15-16 Units sold in Utah
IWISC    DS    H           17-18 Units sold in Wisconsin
IBEGIN   DS    H           19-20 Beginning inventory
IPURCH   DS    H           21-22 Purchases throughout year
IQOH     DS    H           23-24 Actual quantity on hand
ICOST    DS    H           25-26 Cost (each) 99V99
ISELL    DS    H           27-28 Sell for (each) 99V99
*****************************************************************
*        Output message definition                              *
*****************************************************************
OMSG     DS    0CL49
         DC    CL39'COGS16A ... Nationwide dollar sales are'
ODOLLARS DC    XL10'4020206B2021204B2020'   BZZ,ZZ9.99
         END   BEGIN
```

16.5. Muestra del Programa: Contribución de California a las ventas de

421 ⟨*COGS16B.MLC* 421⟩≡

```
          PRINT NOGEN
*****************************************************************
*        FILENAME: COGS16B.MLC                                  *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Produce report for COGSWORTH INDUSTRIES      *
*                    California's contribution to sales.        *
*                    This is a modification of COGS13B.MLC and  *
*                    illustrates binary division.               *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
         WTO    'COGS16B ... Begin execution'
⟨search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP 111⟩
         WTO    'COGS16B ... Normal end of program'
         RETURN
*****************************************************************
*        SETUP - Those things which happen one time only,       *
*                 before any records are processed.             *
*****************************************************************
SETUP    EQU    *
         ST     R10,SVSETUP
         OPEN INVENTRY                    Input is EBCDIC, no CR/LF
         OI     REPORT+10,X'08'           PC/370 ONLY - Convert all
*                                         output from EBCDIC to ASCII
         OPEN REPORT
         BAL    R10,HDGS
         BAL    R10,READ
         L      R10,SVSETUP
         BR     R10
*************************************************************
*        HDGS - Print headings.                                 *
*************************************************************
HDGS     EQU   *
         ST    R10,SVHDGS
         PUT   REPORT,HD1
         PUT   REPORT,HD2
         PUT   REPORT,HD3
         PUT   REPORT,HD4
         PUT   REPORT,HD5
```

```
         PUT   REPORT,HD6
         L     R10,SVHDGS
         BR    R10
*****************************************************************
*        PROCESS - Those things which happen once per record.   *
*****************************************************************
PROCESS EQU    *
         ST    R10,SVPROC
         BAL   R10,FORMAT
         BAL   R10,WRITE
         BAL   R10,READ
         L     R10,SVPROC
         BR    R10
*****************************************************************
*        FORMAT - Format a single detail line.                  *
*****************************************************************
FORMAT   EQU   *
         ST    R10,SVFORM
         MVC   OREC,BLANKS
         MVC   ODESC,IDESC
         LH    R3,ICALIF                Determine total units
         AH    R3,IILL                    sold for this product
         AH    R3,IUTAH
         AH    R3,IWISC                 R3 = Nationwide
         LR    R2,R3
         A     R2,TTOTAL                Add nationwide so far
         ST    R2,TTOTAL                  and save it back.
         CVD   R3,DBLWORD               Convert to packed
         ZAP   PK2,DBLWORD                for printing.
         MVC   OTOTAL,=X'40202120'
         ED    OTOTAL,PK2
         LH    R5,ICALIF                R5 = California only
         LR    R2,R5
         A     R2,TCALIF                Add California so far
         ST    R2,TCALIF                  and save it back.
         CVD   R5,DBLWORD               Convert to packed
         ZAP   PK2,DBLWORD                for printing.
         MVC   OCALIF,=X'40202120'
         ED    OCALIF,PK2
         M     R4,=F'1000'              Dividend will be in (R4,R5)
         DR    R4,R3                    Divisor (nationwide) in R3
         CVD   R5,DBLWORD               Quotient is in R5
         SRP   DBLWORD,64-1,5
         ZAP   PK2,DBLWORD
         MVC   OPCT,=X'40202120'
         ED    OPCT,PK2
```

```
                MVI   OPCT+L'OPCT,PERCENT
                MVC   OCRLF,WCRLF              PC/370 only.
                L     R10,SVFORM
                BR    R10
*************************************************************
*       READ - Read a record.                               *
*************************************************************
⟨search..., read INVENTRY into IREC, EOFSW, SVREAD return 314⟩
*************************************************************
*       WRITE - Write a single detail line.                 *
*************************************************************
⟨search..., write REPORT from OREC, no line counting, SVWRITE return 106⟩
         *************************************************************
         *       WRAPUP - Those things which happen one time only,   *
         *                after all records have been processed.     *
         *************************************************************
         WRAPUP EQU   *
                ST    R10,SVWRAP
                PUT   REPORT,HD6
                MVC   OREC,BLANKS
                MVC   ODESC(6),=C'TOTALS'
                L     R3,TTOTAL                R3 = Nationwide total
                CVD   R3,DBLWORD               Convert to packed
                ZAP   PK2,DBLWORD                for printing.
                MVC   OTOTAL,=X'40202120'
                ED    OTOTAL,PK2
                L     R5,TCALIF                R5 = California only
                CVD   R5,DBLWORD               Convert to packed
                ZAP   PK2,DBLWORD                for printing.
                MVC   OCALIF,=X'40202120'
                ED    OCALIF,PK2
                M     R4,=F'1000'              Dividend will be in (R4,R5)
                DR    R4,R3                    Divisor (nationwide) in R3
                CVD   R5,DBLWORD               Quotient is in R5
                SRP   DBLWORD,64-1,5
                ZAP   PK2,DBLWORD
                MVC   OPCT,=X'40202120'
                ED    OPCT,PK2
                MVI   OPCT+L'OPCT,PERCENT
                MVC   OCRLF,WCRLF              PC/370 only.
                BAL   R10,WRITE
                CLOSE INVENTRY
                CLOSE REPORT
                WTO   'COGS16B ... Sales report on REPORT.TXT'
                L     R10,SVWRAP
                BR    R10
```

```
*****************************************************************
*        Literals, if any, will go here                         *
*****************************************************************
         LTORG
*****************************************************************
*        File definitions                                       *
*****************************************************************
INVENTRY DCB   LRECL=28,RECFM=F,MACRF=G,EODAD=ATEND,
               DDNAME='COGS.BIN'
REPORT   DCB   LRECL=62,RECFM=F,MACRF=P,
               DDNAME='REPORT.TXT'
*****************************************************************
*        RETURN ADDRESSES                                       *
*****************************************************************
```

⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩

```
*****************************************************************
*        Miscellaneous field definitions                        *
*****************************************************************
WCRLF    DC    X'0D25'                  PC/370 ONLY - EBCDIC CR/LF
EOFSW    DC    CL1'N'                   End of file? (Y/N)
BLANKS   DC    CL62' '
TCALIF   DC    F'0'                     Grand total for Calif
TTOTAL   DC    F'0'                     Grand total nationwide
DBLWORD DC     D'0'
PK2      DC    PL2'0'
PERCENT EQU    C'%'
*****************************************************************
*        Input record definition                                *
*****************************************************************
         DS    0H                       Force halfword alignment
IREC     DS    0CL28          1-28 Inventory record
IDESC    DS    CL10           1-10 Product description
ICALIF   DS    H             11-12 Units sold in Calif
IILL     DS    H             13-14 Units sold in Illinois
IUTAH    DS    H             15-16 Units sold in Utah
IWISC    DS    H             17-18 Units sold in Wisconsin
IBEGIN   DS    H             19-20 Beginning inventory
IPURCH   DS    H             21-22 Purchases throughout year
IQOH     DS    H             23-24 Actual quantity on hand
ICOST    DS    H             25-26 Cost (each) 99V99
ISELL    DS    H             27-28 Sell for (each) 99V99
*****************************************************************
*        Output (line) definition                               *
*****************************************************************
OREC     DS    0CL62           1-62
ODESC    DS    CL10            1-10 Product description
```

```
                DS    CL7             11-17
          OTOTAL DS    CL4             18-21 Units sold Nationwide
                DS    CL9             22-30
          OCALIF DS    CL4             31-34 Units sold in Calif
                DS    CL8             35-42
          OPCT   DS    CL4             43-46 Percent sales from Calif
                DS    CL14            47-60
          OCRLF  DS    CL2             61-62 PC/370 only - CR/LF
*************************************************************
       *       Headings definitions                          *
       *************************************************************
       HD1     DS    0CL62
               DC    CL60'              COGSWORTH INDUSTRIES      '
               DC    XL2'0D25'
       HD2     DS    0CL62
               DC    CL60'       California''s Contribution to Sales'
               DC    XL2'0D25'
       HD3     DS    0CL62
               DC    CL60' '
               DC    XL2'0D25'
       HD4     DS    0CL62
               DC    CL40'              Nationwide   California   '
               DC    CL20'Percent of'
               DC    XL2'0D25'
       HD5     DS    0CL62
               DC    CL40' Product         Sales        Sales     '
               DC    CL20' National '
               DC    XL2'0D25'
       HD6     DS    0CL62
               DC    CL40'-----    -----   -----   '
               DC    CL20'-----'
               DC    XL2'0D25'
               END   BEGIN
```

16.6. Ejercicios

1. Verdadero o falso.

   a) es incluso un par impar de registros. F T (R4, R3)

   b) Todas las instrucciones de multiplicación binaria utilizar al menos un registro. F T

   c) El primer operando de una instrucción M debe especificar el registro de un extraño F T incluso par-impar.

d) El primer operando de una instrucción debe especificar la MH, incluso registro de un F T incluso par-impar.

e) A raíz de una instrucción M, el producto va a ocupar incluso un par impar de F T registros.

f) Tras una instrucción MH, el producto va a ocupar una media palabra. F T

g) Es imposible para multiplicar una media palabra por un fullword con el producto F T ocupando la media palabra.

h) Al realizar la división binaria, en previsión de redondeo, multiplicar el F T incluso-extraña pareja que contiene el dividendo por una potencia de diez.

i) Después de M o D, el producto o dividendos deben ser convertidos a decimal empaquetado F T con el fin de mostrar correctamente su valor.

j) Si el dividendo es en un registro y el divisor se encuentra en una media palabra, la instrucción de DH F T puede ser utilizado.

k) La instrucción HD utiliza un total de tres registros. F T

l) La instrucción de RM utiliza un total de dos registros. F T

m) Tras la instrucción de D, el resto será en el registro de números pares F T y el cociente será en el registro de número impar.

2. Habida cuenta de las definiciones de los campos siguientes:

```
H1 DC H'25 '
H2 DC H'8 '
H3 DC H'0 '
F1 DC F'6 '
F2 DC F'3 '
F3 DC F'0 '
```

Encontrar el error (sólo una) en cada uno de los siguientes:

a) * Multiplicar F1 por F2 dando F3

```
L R5, F1
M R5, F2
ST R5, F3
```

b) * Multiplicar F1 por F2 dando F3

```
L R5, F1
L R6, F2
M R4, R6
ST R5, F3
```

c) * Multiplicar F1 por F2 dando F3

```
L R5, F1
M R4, F2
ST R4, F3
```

d) * Multiplicar H1 por H2 dando H3

```
LH R4, H1
MH R4, H2
STH R5, H3
```

e) * Multiplicar H1 por H2 dando H3

```
LH R3, H1
M R3, H2
STH R3, H3
```

f) * Multiplicar F1 por H2 dando F3

```
LH R3, F1
MH R3, H2
ST R3, F3
```

3. Habida cuenta de las definiciones de los campos siguientes:

```
H1 DC H'25 '
H2 DC H'8 '
H3 DC H'0 '
F1 DC F'6 '
F2 DC F'3 '
F3 DC F'0 '
```

Encontrar el error (sólo una) en cada uno de los siguientes:

a) * Divide F1 por F2, cociente en la F3

```
L R5, F1
M R4 = f'1 '
D R4, F2
ST R4, F3
```

b) * Divide F1 por F2, cociente en la F3

```
L R4, F1
M R4 = f'1 '
L R6, F2
DR R4, R6
ST R5, F3
```

c) * Divide H1 por H2, cociente en H3

```
LH R5, H1
M R4 = H'1 '
LH R6, H2
DR R4, R6
STH R5, H3
```

d) * Divide H1 por F2, cociente en H3

```
LH R3, H1
M R2 = f'1 '
D R2, F2
ST R3, H3
```

4. Habida cuenta de las definiciones de los campos siguientes:

```
F1 DC F'16 '
F2 DC F'8 '
H1 DC H'4 '
H2 DC F'3 '
DBL DC D'0 '
PK3 DC PL3'0 '
```

Suministro de las instrucciones para realizar cada una de las siguientes. Mostrar todos los resultados intermedios. Comenzar con nuevos datos cada vez.

a) Multiplique F1 por F2 dando F2.

b) Multiplique H1 por H2 dar H1.

c) Multiplique F1 por F2 dando H1.

d) Multiplique H2 por F2 dando F1.

e) F1 multiplicar por 2 da F1.

f) H1 multiplicar por 2 dando H1.

g) Divide F1 por F2 dando cociente en la F1.

h) Divide F1 resto por dar H1 en H2.

i) Divide F2 por H2 dando cociente en PK3.

j) Divide H2 por F2 dando resto en PK3.

k) Divide F1 por 5 da cociente en PK3.

l) Divide H1 por 5 da resto en H2.

5. Escribir un programa que lea la versión binaria de la Herramienta de archivo (TOOL.BIN) producidos en el ejercicio 8 del capítulo 14 y crear el informe de marcado se muestra en el ejercicio 8 del capítulo 13. ¿Todos los aritméticas en binario, es decir, el uso envasados sólo los campos que sean necesarias para el comando ed.

Capítulo 17

A nivel de bits de Operaciones

Objetivos Al término de este capítulo usted será capaz de:

- Describa lo que se entiende por la o, y, y en exclusiva o de las operaciones,
- Utilice el OI y las instrucciones para activar el OC bits,
- Usar la NI y las instrucciones de NC para desactivar bits,
- Usar la XI y las instrucciones para cambiar XC bits,
- Utilice la instrucción TM para probar bits,
- Utilice el NI OI y las instrucciones para cambiar el caso de una carta,
- Usar la XI y las instrucciones de XC para el cifrado de datos,
- Utilice la instrucción XC para cambiar los campos,
- Utilice el SLL SRL y las instrucciones para cambiar los bits en un registro,
- Utilice el SLL SRL y las instrucciones para multiplicar un registro por una potencia de dos,
- Utilice PC/370 's SVC 18 para acceder a la fecha y hora del sistema.

Introducción En este capítulo veremos algunos de los SYSTEM/370 's operaciones a nivel de bits. La mayoría de estos son bastante especializados: no se necesitan muy a menudo, pero cuando usted los necesita simplemente no hay sin obtener por ellos. En particular, se examinará la OI, NI, XI, OC, Carolina del Norte, XC, TM, SRL, y SLL instrucciones.

17.1. La O, Y, y exclusiva o de operaciones

Por ahora sabemos que cada byte se compone de ocho bits o dígitos binarios. Hay tres a nivel de bits operaciones. Estas operaciones se conocen como O, Y, y exclusiva o. Cada una de estas operaciones de compara bits correspondientes de cada uno de los dos operandos. Cualquier bit del operando destino puede ser cambiado como resultado de la comparación. El resultado dependerá de la tabla de verdad siguiente:

```
The result is 1  The result is 1    The result is 1
           if either bit    if both bits   if exactly one bit
                 is 1            are 1            is 1
Operand 1 0     0    1   1 0     0   1    1   0   0    1   1
Operand 2 0     1    0   1 0     1   0    1   0   1    0   1
Result    0     1    1   1 0     0   0    1   0   1    1   0
```

El almacenamiento correspondiente-y-Inmediata (SI), las instrucciones son (o inmediato), NI (y OI inmediata) y XI (en exclusiva o inmediato).

17.2. El OI, NI, y XI de Instrucciones

En el siguiente análisis se hará referencia a las posiciones de bits por número: la norma es número de bits de De izquierda a derecha, comenzando con cero.

```
0 1 2 3 4 5 6 7
```

Ejemplo#1: Gire a la izquierda-la mayoría de los bits en el primer byte de la FLD.

```
OI FLD,X'80' or OI FLD,B'10000000'
```

Ejemplo#2: Conecte un bit de FLD. Todos los otros bits mantienen sin cambios.

```
OI FLD,X'40' or OI FLD,B'01000000'
```

Pero tenga en cuenta ...

```
X'99' = B'10011001' = C'r'
```

y ...

```
X'D9' = B'11011001' = C'R'
```

Así Ejemplo # 2 ilustra cómo podemos cambiar una letra en minúsculas a mayúsculas!

Ejemplo#3: Conecte bits cero, uno, dos, y tres de FLD.

```
OI FLD,X'F0' or OI FLD,B'11110000'
```

Así Ejemplo 3 ilustra un método por el cual podemos quitar el signo de un después de un número UNPK.

intenta 1. Escriba las instrucciones para girar a la derecha-la mayoría de los bits en el primer byte de FLD.

2. Escriba las instrucciones para activar bits de dos y tres del último byte de X. (Uso de la longitud de operador para indicar el último byte de X.)

3. Ejemplo # 2 muestra cómo se puede cambiar una letra minúscula a mayúscula. ¿Qué pasa si el byte en el cuestión contiene un número, es decir, X'F0 a través de X'F9? ¿Qué efecto, si lo hubiere, el OI la instrucción, como se muestra en la que han byte?

4. Ejemplo # 3 muestra cómo se puede quitar el signo de un número después de una UNPK. Qué instrucción hicimos antes de su uso para hacer esto? ¿Cuál es la longitud de estas instrucciones? ¿Por qué podría este método (OI) se prefiere en la otra?

Ejemplo#4: Apague los bits de la izquierda y la derecha de la FLD.

```
NI FLD,X'7E' or NI FLD,B'01111110'
```

Ejemplo#5: Apague poco cuatro de FLD

```
OI FLD,X'F0' or OI FLD,B'11110000'
```

Pero tenga en cuenta ...

```
X'FF' = B'11111111' = 255
```

y ...

```
X'08' = B'00001000' = 8
```

y ...

```
X'F7' = B'11110111' = 247
```

Así que también podría usar:

```
NI FLD,255-8        or
NI FLD,X'FF'-X'08'  or
NI FLD,ALLBITS-BIT4
```

donde

```
ALLBITS EQU X'FF'
```

y

```
BIT4 EQU X'08'
```

Ejemplo#6: Apaga un bit de FLD. Todos los otros bits mantienen sin cambios. (Este es el reverso de Ejemplo # 2 arriba).

```
NI FLD,B'10111111'   or
NI FLD,X'BF'         or
NI FLD,ALLBITS-X'40'
```

Así Ejemplo # 6 ilustra cómo podemos cambiar una letra en mayúsculas a minúsculas!

Ejemplo#7: Encienda poco siete de los byte más a la derecha de FLDB, un campo de tres Bolsas de bytes, si se trata de fuera, de lo contrario lo apaga.

```
XI FLDB+2,X'01'
```

Cambiar el valor de un campo de esta manera (es decir, el encendido, si fuera, o de transformarla apagado si el) se refiere a veces como cambios. Pero tenga en cuenta ...

```
X'0C' = B'0000110 0' and...
X'0D' = B'0000110 1'
```

Recuerde que C representa un signo positivo en una serie de pic nic, y D representa un signo negativo en una serie de pic nic, y vemos que si el último byte XI, con X'01 ', que "cambiar" entre C y D.

Así Ejemplo # 7 ilustra cómo podemos cambiar el signo de un número de pic nic!

intenta

1. 5. Escriba las instrucciones para desactivar el botón derecho del bit más en el primer byte de FLD.
2. 6. Escriba las instrucciones para desactivar los bits de dos y tres del último byte de X. (Uso de la longitud de operador para indicar el último byte de X.)
3. 7. Ejemplo # 6 se muestra cómo podemos cambiar una letra en mayúsculas a minúsculas. ¿Qué pasa si el byte en cuestión contiene un número, es decir, X'F0 a través de X'F9? ¿Qué efecto, si lo hubiere, la NI la instrucción, como se muestra en la que han byte?
4. 8. Ejemplo # 7 muestra cómo podemos cambiar el signo de un número de pic nic. Se podría lograr lo mismo al multiplicar el número por -1. Dado PK3 contiene X'00012D ', el uso tanto métodos para cambiar el signo.

5. 9. Refiérase a la pregunta anterior. ¿Por qué puede este método (XI), se prefiere en la otra (MP)? Sugerencia: ¿Qué pasa si queremos cambiar el signo de PK3 que contiene X'12345D?

* * * * * * * * * * * * * * * * * * * *

La instrucción XI tiene una curiosa propiedad de que, si un campo es XI'ed con un valor, y la resultante campo es XI'ed con el mismo valor, el campo regresa a su valor original. Esta propiedad es útil en los programas de cifrado como se demuestra en el siguiente ejemplo. la letra "R" con el carácter '+':

Ejemplo#8: el resultado ('p') con el carácter '+'

```
          C'R' = X'D9' = B'11011001'
          C'+' = X'4E' = B'01001110'
                  result = B'10010111' = X'97' = C'p'
  the result ('p') with the character '+' again:
XI          C'p' = X'97' = B'10010111'
          C'+' = X'4E' = B'01001110'
                  result = B'11011001' = X'D9' = C'R'
```

Vemos el resultado («R») es el valor original. Esta propiedad es útil en el cifrado programas!

intenta

1. 10. la letra 'H' con el carácter '$'. XI el resultado con el carácter '$' de nuevo. Mostrar la XI los resultados intermedios.

2. 11. XI de la letra «S» con el carácter '#'. XI el resultado con el carácter '#' de nuevo. Mostrar la los resultados intermedios.

17.3. The OC, Carolina del Norte, y XC Instrucciones

Hay un SS correspondiente (Storage-to-Storage) de instrucciones para cada uno de los anteriores SI instrucciones: son OC, NC, y XC. La función es el mismo que con las instrucciones anteriores, pero en cada caso el segundo parámetro es un campo (o literal) en lugar de un valor inmediato.

Ejemplo # 9: las letras "PR" con los personajes'+;'. (Esto puede ser considerado como XC de cifrado.)

```
C'PR' = X'D7D9' = B'1101011111011001'
C'+;' = X'4E5E' = B'0100111001011110'
          result = B'1001100110000111'
                 = X'9987' = C'rg'
```

el resultado ("RG") con los caracteres '+,' de nuevo. (Esto puede ser considerado como XC descifrado.)

```
C'rg' = X'9987' = B'1001100110000111'
C'+;' = X'4E5E' = B'0100111001011110'
         result = B'1101011111011001'
                = X'D7D9' = C'PR'
```

Vemos el resultado ("RP") es el valor original!

Ejemplo#10: Si tiene un valor FLDA DS CL1 X'10110110 ' y tiene un valor FLDB DS CL1 X'11010010 '

```
                gives FLDA = X'01100100'
XC FLDA,FLDB                (using the new FLDA)
XC FLDB,FLDA                gives FLDB = X'10110110'
               (using the new FLDB)
XC FLDA,FLDB                gives FLDA = X'11010010'
```

Pero tenga en cuenta: FLDA ahora es igual a la "original" FLDB y FLDB ahora es igual a la "original" FLDA!

Así Ejemplo # 10 ilustra cómo se puede utilizar la instrucción de XC de "swap" de la los valores en dos campos. (Los dos campos deben ser del mismo tamaño.)

intenta

1. 12. Teniendo en cuenta WK2 DC CL2'HS. XC el WK2 campo con los caracteres "$ #" dos veces. Mostrar el resultado después de cada uno de XC.

2. 13. Dado A CL2'PJ DC y B DC CL2'B4. Utilice la instrucción XC para intercambiar A y B. Mostrar todos los resultados intermedios.

3. 14. Dada X DC CL2'R2. Determinar los resultados de XC X, X. ¿Qué puede concluir?

17.4. Manipulación de los registros: El SLL y SRL Instrucciones de

El SLL (Shift lógico de izquierda) y SRL (Shift lógico a la derecha) las instrucciones son similares a la SRP (Shift Bolsas y Ronda), la instrucción, salvo que, mientras que la instrucción cambió los dígitos de un pic nic número a la izquierda oa la derecha, estas instrucciones de desplazamiento los bits en un registro a la izquierda oa la derecha. Recall de que el efecto de la SRP fue multiplicar o dividir el número de envasado por alguna potencia de diez. Del

mismo modo, el resultado de la SLL es multiplicar el valor de un registro por alguna potencia de dos, y el resultado de la SRL es dividir el valor de un registro por alguna potencia de dos. Por ejemplo:

Ejemplo#11:

```
                                                0000 0100
                0000 0000 0000 0000 0000 0000
LA  R3,4 R3=                                                    4
                    00        00        00        04
                                                0010 0000
                0000 0000 0000 0000 0000 0000
SLL R3,3                                                       32
                    00        00        00        20
                                                0000 1000
                0000 0000 0000 0000 0000 0000
SRL R3,2                                                        8
                    00        00        00        08
```

La instrucción SLL por encima de los cambios en el registro de todos los bits 3 a la izquierda tres posiciones. El efecto neto es multiplicar ese registro por 23 u 8, que da 4 * 8 = 32. Los cambios de instrucción SRL todos los bits en el registro de 3 a la derecha 2 posiciones. El efecto neto es multiplicar ese registro por 2-2, o dividir por 22, o 4, dando 32 / 4 = 8.

intenta

1. 15. Ejecute las siguientes instrucciones. Mostrar todos los resultados intermedios.

```
LH  R4,=H'48'
SRL R4,4
SLL R4,2
```

2. 16. Ejecute las siguientes instrucciones. Mostrar todos los resultados intermedios.

```
LH  R4,=H'20'
SRL R4,3
SLL R4,3
```

17.5. Programa de ejemplo: a nivel de bits de Operaciones

El siguiente programa, BITOPS.MLC, demostrará la mayoría de los ejemplos expuestos más arriba. El la salida de la ejecución del programa sigue el listado

de código fuente.

436 ⟨ff 40⟩+≡ ◁395 440▷

```
         PRINT NOGEN
*****************************************************************
*        FILENAME: BITOPS.MLC                                   *
*        AUTHOR : Bill Qualls                                   *
*        SYSTEM : PC/370 R4.2                                   *
*        REMARKS : Demonstrate bit-level operations.            *
*****************************************************************
         START 0
         REGS
BEGIN    BEGIN
*
         WTO   'EXAMPLE #2 - Demonstrate use of OI to change'
         WTO   'lower case letter to upper case'
         WTO   LOWER
         OI    LOWER,X'40'
         WTO   LOWER
*
         WTO   'EXAMPLE #3 - Demonstrate use of OI to remove'
         WTO   'the sign from a number following an UNPK'
         UNPK UNPACKED,=P'-12345'
         WTO   UNPACKED
         OI    UNPACKED+L'UNPACKED-1,X'F0'
         WTO   UNPACKED
*
         WTO   'EXAMPLE #6 - Demonstrate use of NI to change'
         WTO   'upper case letter to lower case'
         WTO   UPPER
         NI    UPPER,ALLBITS-X'40'
         WTO   UPPER
*
         WTO   'EXAMPLE #7 - Demonstrate use of XI to ''toggle'''
         WTO   'the sign of a packed number'
         MVC   EDITED,MASK
         ED    EDITED,POSITIVE
         WTO   EDITED
         XI    POSITIVE+L'POSITIVE-1,X'01'
         MVC   EDITED,MASK
         ED    EDITED,POSITIVE
         WTO   EDITED
         XI    POSITIVE+L'POSITIVE-1,X'01'
         MVC   EDITED,MASK
         ED    EDITED,POSITIVE
         WTO   EDITED
```

```
*
         WTO   'EXAMPLE #8 - Demonstrate use of XI for'
         WTO   'encryption: once to encrypt, once to decrypt.'
         WTO   CRYPT1
         XI    CRYPT1,C'+'         encrypt
         WTO   CRYPT1
         XI    CRYPT1,C'+'         decrypt
         WTO   CRYPT1
*
         WTO   'EXAMPLE #9 - Demonstrate use of XC for'
         WTO   'encryption: once to encrypt, once to decrypt.'
         WTO   CRYPT2
         XC    CRYPT2,=C'+;' encrypt
         WTO   CRYPT2
         XC    CRYPT2,=C'+;' decrypt
         WTO   CRYPT2          *
      WTO    'EXAMPLE #10 - Demonstrate use of XC to swap'
      WTO    'two values'
      WTO    BOTH
      XC     FLDA,FLDB
      XC     FLDB,FLDA
      XC     FLDA,FLDB
      WTO    BOTH
*
      WTO    'EXAMPLE #11 - Demonstrate that SLL is same as'
      WTO    'multiplying a register by a power of two, and'
      WTO    'that SLR is same as dividing by a power of two.'
      LA     R3,4            We begin with 4
      CVD    R3,DBLWORD
      MVC    EDITED,MASK
      ED     EDITED,DBLWORD+5
      WTO    EDITED
      SLL    R3,3            Multiply 4 by 2^3, or 8, giving 32
      CVD    R3,DBLWORD
      MVC    EDITED,MASK
      ED     EDITED,DBLWORD+5
      WTO    EDITED
      SRL    R3,2            Divide 32 by 2^2, or 4, giving 8
      CVD    R3,DBLWORD
      MVC    EDITED,MASK
      ED     EDITED,DBLWORD+5
      WTO    EDITED
*
      RETURN
*
      LTORG
```

```
*
DBLWORD  DC     D'0'
MASK     DC     XL7'40202020212060'
EDITED   DC     CL7' '
POSITIVE DC     PL3'+6789'
UNPACKED DC     CL5' '
LOWER    DC     CL1'r'    Lower case letter 'r'
UPPER    DC     CL1'T'    Upper case letter 'T'
ALLBITS  EQU    X'FF'
CRYPT1   DC     CL1'R'
CRYPT2   DC     CL2'PR'
BOTH     DS     0CL9
FLDA     DC     CL3'123'
         DC     CL3' '
FLDB     DC     CL3'AbC'
         END
```

```
A:\MIN>bitops
EXAMPLE #2 - Demonstrate use of OI to change
lower case letter to upper case
r
R
EXAMPLE #3 - Demonstrate use of OI to remove
the sign from a number following an UNPK
1234N
12345
EXAMPLE #6 - Demonstrate use of NI to change
upper case letter to lower case
T
t
EXAMPLE #7 - Demonstrate use of XI to 'toggle'
the sign of a packed number
   6789
   6789-
   6789
EXAMPLE #8 - Demonstrate use of XI for
encryption: once to encrypt, once to decrypt.
R
p
R
EXAMPLE #9 - Demonstrate use of XC for
encryption: once to encrypt, once to decrypt.
PR
rg
PR
EXAMPLE #10 - Demonstrate use of XC to swap
two values
123     AbC
AbC     123
EXAMPLE #11 - Demonstrate that SLL is same as
multiplying a register by a power of two, and
that SLR is same as dividing by a power of two.
      4
     32
      8
```

intenta

1. 17. Escribir un programa similar para demostrar sus respuestas a todos los anteriores intenta Ejerce.

17.6. Programa de ejemplo: Acceso a la fecha y la hora

El siguiente programa, DATE370.MLC, utiliza varias de estas instrucciones para recuperar la fecha del sistema y la tiempo. Este programa hace uso de la llamada supervisor de 18, que devuelve el tiempo en el registro 0, el año de (con el siglo) en el registro 1, y el día, mes y día de la semana en el indicador de registro 2. Estos a continuación, los registros son manipulados con el fin de devolver la fecha y la hora en un formulario estándar. Significativa comentarios se han utilizado throughtout. De particular interés es el medio por el cual el SLL y la Instrucciones SRL se utilizan juntos para aislar una parte de un registro. Nota: SVC 18 se discute en PC/370 's documentación. El uso de la VPC 18 para obtener la fecha y hora del sistema es único para PC/370.

440 ⟨ff 40⟩+≡ ◁436

```
        PRINT NOGEN
*****************************************************************
*       FILENAME: DATE370.MLC                                   *
*       AUTHOR : Bill Qualls                                    *
*       SYSTEM : PC/370 R4.2                                    *
*       REMARKS : Demonstrate date/time functions in PC/370.    *
*****************************************************************
        START 0
        REGS
BEGIN   BEGIN
        WTO   MESSAGE                   (Before)
*
        SVC   18
*                                       Supervisor call 18 returns
*                                       time in R0; year with century
*                                       in R1; day, month, and day of
*                                       week in R2.
*
        LR    R3,R0                     Put time in R3
        SRL   R3,24                     hhmmssxx becomes 000000hh
        CVD   R3,DBL                    Hours only
        UNPK TIME(2),DBL                Move to output
        OI    TIME+1,X'F0'              Remove sign
*
        LR    R3,R0                     Put time in R3
        SLL   R3,8                      hhmmssxx becomes mmssxx00
        SRL   R3,24                     mmssxx00 becomes 000000mm
        CVD   R3,DBL                    Minutes only
        UNPK TIME+3(2),DBL              Move to output
        OI    TIME+4,X'F0'              Remove sign
*
```

```
            LR    R3,R0                     Put time in R3
            SLL   R3,16                     hhmmssxx becomes ssxx0000
            SRL   R3,24                     ssxx0000 becomes 000000ss
            CVD   R3,DBL                    Seconds only
            UNPK TIME+6(2),DBL              Move to output
            OI    TIME+7,X'F0'              Remove sign
*
            LR    R3,R0                     Put time in R3
            SLL   R3,24                     hhmmssxx becomes xx000000
            SRL   R3,24                     xx000000 becomes 000000xx
            CVD   R3,DBL                    Hundredths of seconds only
            UNPK TIME+9(2),DBL              Move to output
            OI    TIME+10,X'F0'             Remove sign
*
            CVD   R1,DBL                    Year with century
            UNPK DATE+6(4),DBL              Move to output
            OI    DATE+9,X'F0'              Remove sign
*
            LR    R3,R2                     Put date in R3
            SRL   R3,24                     mmddww00 becomes 000000mm
            CVD   R3,DBL                    Month only
            UNPK DATE(2),DBL                Move to output
            OI    DATE+1,X'F0'              Remove sign
*
        LR     R3,R2              Put date in R3
        SLL    R3,8               mmddww00 becomes ddww0000
        SRL    R3,24              ddww0000 becomes 000000dd
        CVD    R3,DBL             Day of month only
        UNPK   DATE+3(2),DBL      Move to output
        OI     DATE+4,X'F0'       Remove sign
*
        LR     R3,R2              Put date in R3
        SLL    R3,16              mmddww00 becomes ww000000
        SRL    R3,24              ww000000 becomes 000000ww
        MH     R3,=H'3'           Each day of week is 3 long
        A      R3,=A(DOWTBL)      Displacement into table
        MVC    DOW,0(R3)          Move to output
*
        WTO    MESSAGE            (After)
*
        RETURN
*
        LTORG
*
MESSAGE DS     0CL71
        DC     CL18'DATE370...Time is '
```

```
TIME    DC      CL11'hh:mm:ss.xx'
        DC      CL11'...Date is '
DATE    DC      CL10'mm/dd/yyyy'
        DC      CL18'...Day of week is '
DOW     DC      CL3'ddd'
*
DBL     DS      D
DOWTBL  DC      C'SunMonTueWedThuFriSat'
*
        END
```

```
A:\MIN>date370
 DATE370...Time is hh:mm:ss.xx...Date is mm/dd/yyyy...Day of week is ddd
 DATE370...Time is 08:49:54.44...Date is 01/06/1994...Day of week is Thu
```

* * * * * * * * * * * * * * * * * * * *

La capacidad de activar o desactivar bits seleccionados significa que podemos usar bits como interruptores. En particular, cualquier condición binaria (una condición con sólo dos estados posibles) puede ser representado con un solo bit en lugar de un byte entero. Esto puede resultar en un ahorro considerable de espacio en disco y el tiempo y el costo de las telecomunicaciones. Algunos ejemplos de las condiciones binario son:

```
OFTEN          CAN ALSO BE CONDITION
                 REPRESENTED AS   REPRESENTED AS
                           'F' = Female    0 = Female
 GENDER
                           'M' = Male      1 = Male
                           'N' = No        0 = No TENURED
                           'Y' = Yes       1 = Yes
                           'C' = Check     0 = Check
 CHECKING ACCOUNT
                           'D' = Deposit   1 = Deposit
 TRANSACTION TYPE
                           ' ' = No        0 = No
 OUT OF STOCK
                           'X' = Yes       1 = Yes
                           ' ' = No        0 = No
 MARKED FOR DELETION
                           '*' = Yes       1 = Yes
 (as used in dBASE III+)
```

17.7. Bits de cheques: La Instrucción TM

Como ya hemos visto, podemos utilizar la OI, NI, XI y las instrucciones para activar o desactivar bits. De Por supuesto, no nos hacen ningún bien a utilizar un poco como un interruptor, si no también puede probar el valor de ese trozo. El TM (Ensayo con la instrucción Mask) se utiliza para hacerlo. La instrucción TM es un tipo de SI - la instrucción y tiene el campo TM forma, la máscara de

Es inmediatamente seguida por una AC (rama de la condición), normalmente usando uno de los siguientes mnemónicos extendido:

```
MNEMONIC MEANING            BC EQUIVALENT
BO                          BC 1,label          Branch if Ones
BM                          BC 4,label          Branch if Mixed
BZ                          BC 8,label          Branch if Zeros
```

```
BNO                         BC 14,label         Branch if Not Ones
BNM                         BC 11,label         Branch if Not Mixed
BNZ                         BC 7,label          Branch if Not Zeros
```

Ejemplo#12: Si los bits primero y tercero de FLDA estás, y luego apague el tercer bit. De lo contrario, activar el bit de sesiones.

```
          TM  FLDA,B'10100000'
          BO  OFF3RD
          OI  FLDA,B'00000010'
          B   DONE
OFF3RD    EQU *
          NI  FLDA,255-B'00100000'
DONE      EQU *
```

Ejemplo#13: Si el quinto, séptimo u octavo bits de FLDB estás, y luego encender el primer bit.

```
          TM  FLDB,B'00001011'
          BZ  ALLOFF
          OI  FLDB,B'10000000'
ALLOFF    EQU *
```

No hay un equivalente SS a la instrucción TM: sólo se puede probar un byte a la tiempo, y usted puede usar un valor inmediato solamente. Por supuesto, como con todos los SI instrucciones, usted puede usar valores equivalentes. Por ejemplo, para probar la igualdad de género masculino, uno podría código:

```
TM INFO,MALE       where
INFO    DS   CL1
MALE    EQU  X'80' First bit indicates gender
CITIZEN EQU  X'40' Second bit indicates citizenship
*                  Other bits unused at this time
```

intenta

1. 18. Si el último bit del primer byte de A está encendido, y el primer bit del último byte de B está apagado, luego girar a la en el primer bit del primer byte de la C.

2. 19. Habida cuenta INFO, MALE, y ciudadano como se define anteriormente, y SWITCH DC CL1 '. Si INFO indica un mujer ciudadana, mover 'Y' para cambiar. De lo contrario. mover 'N' para cambiar.

17.8. Ejercicios

1. Verdadero o falso.

   Un hecho y

   ```
   B DC CL2'5H '... DC CL2'IQ '
   ```

   a) Para activar el último bit en el último byte de A, dejando a todos los otros bits sin cambios, nos F T código de OI A +1, X'08 '

   b) Para desactivar el primer bit en el último byte de A, dejando a todos los otros bits sin cambios, F T que el código NI A +1, X'80 '

   c) Para desactivar el bit de la izquierda en el primer byte de A si está encendido, y que a su vez en esa F T si está fuera de bits, código XI A, X'80 '

   d) Para cambiar la 'Q' de la A a minúsculas. que el código NI A +1, B'10111111 ' F T

   e) Para intercambiar A y B que el código XC A, B tres veces. F T

   f) El valor de B puede haber sido un resultado de UNPK B, PK3 donde PK3 es un pic nic F T número que contiene 158.

   g) OI B + L'B-1, X'F0 'dará B igual a CL2'58' F T

   h) Dado TM A, X'C0 y BZ PASE la rama de que se adopten. F T

   i) Dado TM B +1, B'10000000 'y BO PASE la rama que se adopten. F T

   j) OC A, B da un X'3C10 iguales ». F T

   k) La instrucción SRL se usa para multiplicar un registro por una potencia de 10. F T

   l) Habida cuenta de registro de 4 contiene 10. Después SLL R4, 3 seguido SRL R4, 3 registro 4 F T todavía contiene 10.

   m) Habida cuenta de registro de 4 contiene 10. Después SRL R4, 3 seguido por SLL R4, 3 registro 4 F T todavía contiene 10.

2. Complete los siguientes cuadros:

   ```
                                  EXCLUSIVE
                                      OR
   Operand 1 0 1 0 1 0 1 0 1 0 1   0 1
   Operand 2 0 0 1 1 0 0 1 1 0 0   1 1
   Result
   ```

3. Suministro de los bits de la máscara y la bits resultantes de FLD:

   a) OI FLD, FLD X'FC '1 1 0 0 0 1 1 0 MÁSCARA FLD

   b) NI FLD, FLD X'E4 '1 1 0 0 0 1 1 0 MÁSCARA FLD

c) XI FLD, FLD X'7A '1 1 0 0 0 1 1 0 MÁSCARA FLD

d) OI FLD, FLD X'B3 '1 1 0 0 0 1 1 0 MÁSCARA FLD

e) NI FLD, FLD X'A6 '1 1 0 0 0 1 1 0 MÁSCARA FLD

f) XI FLD, FLD X'AA '1 1 0 0 0 1 1 0 MÁSCARA FLD

g) OI FLD, FLD X'0F '1 1 0 0 0 1 1 0 MÁSCARA FLD

4. Teniendo en cuenta los valores especificados para (antes), mascarilla y FLD (después), el suministro de los desaparecidos FLD instrucción.

   a) FLD 1 1 0 0 0 1 1 0 MÁSCARA 1 1 1 0 0 1 0 0 FLD 1 1 0 0 0 1 0 0

   b) FLD 1 0 0 0 1 1 0 1 MÁSCARA 1 1 1 1 1 0 0 1 FLD 1 1 1 1 1 1 0 1

   c) FLD 1 1 0 1 0 0 0 1 MÁSCARA 0 0 0 0 1 1 1 1 FLD 1 1 0 1 1 1 1 0

   d) FLD 1 1 0 1 0 1 1 1 Máscara 1 1 0 1 0 0 1 0 FLD 1 1 0 1 0 1 1 1

   e) FLD 1 1 0 1 0 1 1 1 MÁSCARA 0 0 1 1 0 1 0 0 FLD 0 0 0 1 0 1 0 0

   f) FLD 1 1 0 1 0 1 1 1 MÁSCARA 0 0 1 1 0 1 0 1 FLD 1 1 1 0 0 0 1 0

   g) FLD 0 1 1 0 1 1 0 0 MÁSCARA 1 0 1 0 0 1 1 1 FLD 0 0 1 0 0 1 0 0

5. Escriba el código de BAL para cada uno de los siguientes:

   a) Si los bits de primer, segundo y tercero de FLDA no están en todos, a continuación, girar a la sexta y bits de sesiones.

   b) Si el bit de cuarta o quinta parte de FLDB está apagado, a continuación, encienda el primer bit, de lo contrario apaga el último bit.

6. Sobre la base de nuestro análisis de la utilización de la instrucción de XC, escribir un programa que cifrar un archivo. Escribir otro (similar) programa que descifrar un archivo. Usted puede ya sea duro código de la "clave" en el programa, o leer desde un archivo.

   Nota: Los ejemplos 8 y 9 de este capítulo han sido cuidadosamente elegidos para dar salida con caracteres no imprimibles. Es poco probable que su programa lo hará para todos los caracteres. Del mismo modo, es poco probable que todos los caracteres resultantes de su rutina de cifrado (que trabaja en EBCDIC) tendrá un carácter correspondiente en el juego de caracteres ASCII. Por lo tanto, asegúrese de que su rutina de cifrado escribe un archivo de EBCDIC, y que su programa de descifrado lee un archivo EBCDIC. Para ello, basta con omitir la instrucción OI utilizados antes de la macro OPEN. El uso de EBCDIC en el PC le da un nivel de de cifrado de todos modos! (Esta consideración vs EBCDIC ASCII fue mencionado en el capítulo 14.)

7. en este capítulo para escribir una copia de rutina que devuelve la fecha del sistema en la Utilice DATE370.MLC 'dd / mm / aa' formato (no del siglo). Llame a su DATE370.CPY rutina. Modificar uno de sus los

programas de informe existente para utilizar esta rutina para obtener la fecha del sistema, y que la fecha de impresión en las líneas. (Copia se debatió por primera vez en el capítulo 13.)

8. Utilice DATE370.MLC en este capítulo para escribir una copia de rutina que devuelve la hora del sistema en (formato no centésimas). Llame a su TIME370.CPY rutina. Modificar uno de sus 'hh: mm: ss' los programas de informe existente para utilizar esta rutina para obtener la hora del sistema, y la impresión que el tiempo en las líneas. (Copia se debatió por primera vez en el capítulo 13.)

Capítulo 18

Indices

Índice de cuadros

1.1. Student 17
1.2. Course 17
1.3. Teacher 18
1.4. Offer 18
1.5. Grade 19

Chunks ⟨*ascii* 38⟩
⟨*cap 3* 86⟩
⟨*COGS13A.MLC* 315⟩
⟨*COGS14A.MLC* 353⟩
⟨*COGS14B.MLC* 357⟩
⟨*COGS16A.MLC* 418⟩
⟨*COGS16B.MLC* 421⟩
⟨*COGS7A.MLC* 157⟩
⟨*COGS7B.MLC* 164⟩
⟨*COGS9B.MLC* 211⟩
⟨*ejemplo 1* 41a⟩
⟨*Epilogo* 26a⟩
⟨*ex* 37⟩
⟨*FF* 327⟩
⟨*ff* 40⟩
⟨*Field A* 76a⟩
⟨*Field B* 76b⟩
⟨*Field C* 76c⟩
⟨*Field W* 77⟩
⟨*MOVE2A.MLC, mostrar movimiento de caracteres, version completa* 49⟩
⟨*MOVE2B.MLC, mostrar movimiento de caracteres con =CL1'-', version completa* 50⟩
⟨*OFFER10A.MLC* 223⟩
⟨*OFFER10B.mlc* 232⟩
⟨*OFFER10C.mlc* 241⟩
⟨*out* 54⟩
⟨*programa no.1, version completa* 13⟩

⟨*Prologo* 23a⟩
⟨*search... CHKLNS, LNS gt MAXLNS HDGS, SVCHKLNS* 184⟩
⟨*search..., CHKDEPT, HOLDDEPT vs IDEPT to ENDDEPT,SVCHKDEP* 239b⟩
⟨*search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM+IDEPT, SVCHKSEM* 239a⟩
⟨*search..., CHKSEM, HOLDSEM vs ISEM to ENDSEM, SVCHKSEM* 229⟩
⟨*search..., main loop SETUP,EOFMAST,EOFTRANS to EOJ,PROCESS,WRAPUP* 280⟩
⟨*search..., main loop SETUP,EOFSTUD to EOJ,PROCESS,WRAPUP* 389⟩
⟨*search..., main loop SETUP,EOFSW to EOJ,PROCESS,WRAPUP* 111⟩
⟨*search..., main loop SETUP,EOFTEACH,EOFOFFER to EOJ,PROCESS,WRAPUP* 260⟩
⟨*search..., read INVENTRY into IREC, EOFSW, SVREAD return* 314⟩
⟨*search..., read INVENTRY into IREC, EOFSW,IN++ SVREAD return* 156⟩
⟨*search..., read OFFER into IREC, EOFSW, SVREAD return* 222⟩
⟨*search..., read TEACHERS into IREC, EOFSW, SVREAD return* 112⟩
⟨*search..., setup ascii INVENTRY and REPORT, HDGS, prime READ, SVSETUP return* 206⟩
⟨*search..., SVSETUP,SVHDGS,SVPROC,SVREAD,SVFORM,SVWRITE,SVWRAP* 109⟩
⟨*search..., write REPORT from OREC, LNS++, SVWRITE return* 185⟩
⟨*search..., write REPORT from OREC, no line counting, SVWRITE return* 106⟩
⟨*TEACH11A.MLC* 266⟩
⟨*TEACH15A.MLC* 379⟩
⟨*TEACH1A.MLC, listado de maestros, version completa* 24⟩
⟨*TEACH1B.MLC, listado de maestros sin LF, version completa* 28⟩
⟨*TEACH1C.MLC, listado de maestros a Archivo, version completa* 30⟩
⟨*TEACH1D.MLC, listado de maestros status a consola, version completa* 32⟩
⟨*TEACH1D.MLC, primer WTO* 31a⟩
⟨*TEACH1D.MLC, segundo WTO* 31b⟩
⟨*TEACH1D.MLC, tercer WTO* 31c⟩
⟨*TEACH2A.MLC, area de trabajo* 43a⟩
⟨*TEACH2A.MLC, formateo del reporte* 43b⟩
⟨*TEACH2A.MLC, linea de salida* 42b⟩
⟨*TEACH2A.MLC, listado de maestros con formato, version completa* 45⟩
⟨*TEACH4B.MLC* 107⟩
⟨*TEACH4C.MLC* 113⟩
⟨*TEACH8A.MLC* 187⟩
⟨*TEACHER IREC, registro de entrada* 44⟩
⟨*TEACHERS DCB, EODAD=ATEND* 26c⟩

Ids